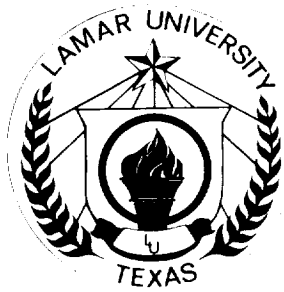
Lamar University

BEAUMONT, TEXAS

TECHNICAL REPORT



COLLEGE OF ENGINEERING

Damage Tolerance in Filament-Wound Graphite/Epoxy Pressure Vessels

FINAL REPORT
NASA Grant NAG 9-698

July 1, 1995

Lamar University
College of Engineering
Beaumont, Texas 77710

William E. Simon, Ph.D., P.E., Principal Investigator
Vinh D. Nguyen, Ph.D., Research Associate
Ravi K. Chenna, Research Assistant

TABLE OF CONTENTS

LIST OF FIGURES

TABLE

TABLES

ABSTRACT

Damage Tolerance

in

Filament-Wound Graphite/Epoxy Pressure Vessels

by

William E. Simon, Ph.D., P.E. Principal Investigator
Vinh D. Nguyen, Ph.D., Research Associate
Ravi K. Chenna, Research Assistant

Graphite/epoxy composites are extensively used in the aerospace and sporting goods industries due to their superior engineering properties compared to those of metals. However, graphite/epoxy is extremely susceptible to impact damage which can cause considerable and sometimes undetected reduction in strength. An inelastic impact model was developed to predict damage due to low-velocity impact. A transient dynamic finite element formulation was used in conjunction with the 3D Tsai-Wu failure criterion to determine and incorporate failure in the material during impact. Material degradation can be adjusted from no degradation to partial degradation to full degradation. The developed software is based on an object-oriented implementation framework called Extensible Implementation Framework for Finite Elements (EIFFE).

CHAPTER 1

Introduction

1.1 Low-Velocity Impact Damage in Composites

Graphite/epoxy composites are used extensively in the aerospace and sports equipment industries because of their high modulus and strength-to-weight ratios, extended fatigue life, and excellent corrosion resistance. These composites are replacing metals in aerospace structures such as aircraft wings and fuselages as well as in sports equipment such as bicycle frames, tennis rackets, and vaulting poles. Graphite/epoxy has been identified as a candidate material for pressure vessels in space applications (Lloyd and Knight, 1986). However, graphite/epoxy is extremely susceptible to impact damage, especially at low impact velocities where internal damage may occur without manifestation on the surface. This particular kind of damage causes considerable reduction in strength and stiffness of the structure (Chang & Choi, 1991; Husman & Whitney, 1975; Jih & Sun, 1993; Poe & Garber, 1987). For instance, experimental work done by Poe and Garber, (1987) on graphite-epoxy laminate cut from a filament-wound composite (FWC) pressure vessel showed 39% reduction in the laminate strength due to low-velocity impact damage.

Several researchers have studied damage behavior of composite laminates undergoing low-velocity impact in the past two decades. These studies examined diverse topics including damage initiation, damage propagation, type of damage, influence of impactor mass and velocity, and laminate lay-up sequence (Kook et al. 1992; Preston & Cook, 1975; Chaturvedi & Sierakowski, 1985). The studies on the effect of stacking

sequence on impact resistance indicated that laminates with more uniformly dispersed ply orientation have greater resistance (Choi et al. 1992). It was also found that impact damage is more sensitive to stacking sequence than to thickness. The damage zones were examined either visually, under an electron microscope, or acoustically using ultrasonic C-scanning and imaging (Chaturvedi & Seirakowski, 1985; Chang & Ketan, 1983). The overall finding was that delamination accompanied by matrix cracking was the major damage mode associated with low-velocity impact. Delamination tends to occur at ply interfaces where the fiber orientation changes (Wu and Springer, 1988). Choi and Chang (1991), found that the delamination area is positively correlated to the impactor energy and recognized that out-of-plane tensile stress was the cause for delamination growth. Matrix cracking, on the other hand, usually occurs at the bottom-most layers. These matrix cracks were also found to initiate delamination at ply interfaces where the fiber orientation changed direction.

1.2. Experimental Studies

Various experimental models including drop-weights (Madan, 1991; Oplinger & Slepetz, 1975), pendulums (Chou, Carleone & Mortimer, 1975; Cook & Preston, 1975), and air guns (Wu & Springer, 1986; Choi & Chang, 1991). Impactors of different shapes and masses were designed to measure the contact force and impact velocities to determine the laminate impact response. Experimental results from both dynamic and static indentation tests suggested that under low-velocity impact conditions, the dynamic impact responses were similar to static indentation responses because the plots obtained for energy absorbed-to-

energy available ratio were similar to the force-indentation plots (Sjöblom et al. 1988). Similar results were also obtained by Kwon and Bhavani (1993) when heavier impactor masses (1 to 10 Kg) with low-velocities (0-3 m/sec) were used.

1.3. Analytical Studies

Simultaneously, analytical models were also being developed to simulate the impact behavior of the composite laminates. These models can be classified as either wave front models or contact models.

1.3.1 Wave Front Models. Wave Front models simulate the impact load using approximation functions and Fourier series to describe the displacement field in the laminate. The equations of motion are derived in terms of these Fourier series and decoupled by transformation to modal coordinates (principal coordinate system). The stress wave patterns, wave surfaces, and wave velocities in the material due to impact can then be obtained by solving the model equations (Kubo & Nelson, 1975; Moon, 1972, 1973; Slepetz & Oplinger, 1975). In these analyses, the loading behavior was simulated using certain approximation functions, whereas in reality, the impact loading behavior and the contact behavior is complicated and cannot be so easily represented.

1.3.2 Contact Models. Contact models use Hertz's contact law as the basis for the computation of contact forces between the impactor and the impacted structure:

$$F = n\alpha^{3/2} \qquad (1.1)$$

where F is the contact force,

α is distance between the centers of gravity of the two bodies, and

$$n = 1.33 * \sqrt{R}\left(\frac{1-\upsilon^2}{E} + \frac{1-\upsilon_p^2}{E_p}\right), \qquad (1.2)$$

E_p = Young's modulus of the projectile,

E = Young's modulus of the target,

ν_p = Poisson's ratio of the projectile,

ν = Poisson's ratio of the target, and

R = projectile radius.

Hertz contact law predicted good results for isotropic materials, but failed for other types of materials. For transversely isotropic targets, Willis (1966) proposed using the transverse modulus of elasticity of the target in equation 1.2. Preston and Cook (1975), Sun (1977) used this approach to compute the impact response of a beam. These researchers solved non-linear integral equations to determine both the contact force and the impact response. While these studies demonstrated the possible application of Hertz's law to impact analysis of composite structures, the results obtained were not close to experimental results, probably due to the assumption that the impact was elastic.

The modified Hertz law could not be successfully implemented for the following reasons:

i) Most laminates could not be represented adequately by a half-space,

ii) It did not account for anisotropic and nonhomogeneous laminate properties.

Furthermore, while Hertz's law assumes perfect elasticity in both the impactor and the target, it was found that permanent indentations

may result at the point of impact. To account for permanent indentations, Crook (1952), proposed the following equation for contact force:

$$F = F_m\left[\frac{\alpha-\alpha_o}{\alpha_m-\alpha_o}\right]^q \tag{1.3}$$

where, F_m = maximum contact force at unloading,

αm = indentation at unloading,

αo = permanent indentation,

q = unloading power.

Yang and Sun (1982) found that indentation exist at very low-velocity impacts and that the loading and unloading behavior changes when maximum indentation exceeds a value called "critical indentation". Contact parameters in the Crook's indentation law were derived by fitting experimental data was fitted for the loading and unloading processes. These indentation laws were successfully used by many researchers to determine the impact response and stress-strain histories (Kook et al. 1992; Chen & Sun, 1985; Sun & Tan, 1985). Tan and Sun (1985) successfully used them in a finite element model to study the dynamic response of a thin graphite/epoxy laminated plate with free-end boundary conditions and small deflections. The results obtained from this analysis agreed with experimental results. Sun & Chen (1985) investigated the influence of effects of inplane prestress on the plate, and the impactor's velocity, mass, and size. Previous studies were based on small-deflection theory which underestimated the magnitude of shear deformation. Later studies revealed that laminates undergo large deflections as well as transverse shear deformation during low-velocity

impact. Shivakumar et al. (1985) and Kook et al. (1992) included the effect of transverse shear deformation in their models. Shivakumar used thick plates ($l/w \leq 12$) in his study, while Kook used higher order shear deformation theory. Results obtained from these studies were close to experimental results.

1.4 Residual Strength Prediction Models

The most popular models available for predicting residual strength in composite structures are empirical. Such empirical models use fracture mechanics concepts to estimate the residual strength by establishing the relationship between residual strength, and impact parameters, such as the impactor velocity, or energy. The strength of any structure depends on its material integrity, i.e., if there is a flaw in the material, the strength of the material decreases. In such models, experimental data for the residual strength and the damage size is first compiled. An empirical relationships is then established between the governing parameters of damage, such as, the impactor velocity or impactor energy and flaw size.

The residual strength of the structure is estimated in conjunction with experimental data and the established relationships. Caprino (1984) used a linear elastic fracture mechanics model to estimate the residual strength of composite laminates as a function of impactor energy. According to fracture mechanics principles, there exists a notch of characteristic length beyond which the structure experiences strength degradation according to the following formula:

$$\frac{\sigma_r}{\sigma_o} = \left(\frac{C_o}{C}\right)^n \qquad (1.4)$$

where, σ_r = residual strength of structure for a notch of length C,

σ_o = residual strength of the structure for a characteristic notch length C_o,

n is the empirical constant.

The damage caused is directly proportional to impact energy, i.e., the higher the impact energy, the larger the damage. This damage may be represented by a notch of an equivalent length determined as one created by the same amount of energy:

$$C = kU^m \tag{1.5}$$

where, C = the equivalent notch length,

k = Constant of proportionality,

U = Energy required to create the notch,

m = Empirical constant.

Using the above relationship, the impactor energy for the characteristic notch can be determined. This equation, when substituted in equation 1.4, yields a relationship between residual strength and impactor energy:

$$\frac{\sigma_r}{\sigma_o} = \left(\frac{U_o}{U}\right)^{mn} \tag{1.6}$$

This model is simple and works well for materials which are rate-insensitive.

Morton and Cantwell (1990), and Husman et al. (1975) represented damage as a sharp crack of equivalent transverse dimension. The residual strength was determined from experimentally established crack length-strength relationships. Poe and Garber (1987) determined the damage depth from analytical methods and represented this damage as a

notch, whose depth was determined empirically to determine the residual strength. The drawback to these methods was that damage was represented as a nonexistent hole or a crack and may not adequately describe the material degradation incurred.

Tian and Swanson (1992) presented an analytical model consisting of damage analysis on a ply-by-ply basis using a 3D finite element model. This model accounted for fiber breakage and delamination as two separate failure modes. Line cracks were modeled to represent the fiber breakage and delaminations. A strain criterion was selected to predict the residual strength. The results obtained from this analysis closely approximated experimental data.

1.5 Damage Characterization

Damage characterization plays a central role in the prediction of residual strength as well as remaining life of composite structures. In the 1980s, the major thrust of research was in the area of damage quantification. Gu and Sun (1983) correlated specimen thickness change and reduction in the specimen's tensile modulus and strength. Ketan et al. (1983) concentrating on the point impact damage to sheet molding compound (SMC) panels, showed that the damage area is the primary parameter in characterizing the damage, rather than the indentation depth, or reduction of thickness. Gu and Sun (1987) proposed a semiempirical approach to predict the damage zone in SMC panels using the strain energy density function. The method is based on the assumption that composite failure is caused by excessive strain energy. A weighted strain energy function was proposed:

$$U = U_V + \beta U_S \tag{1.7a}$$

where U_V = dilatational energy,

U_S = distortional energy,

β = is a weighting coefficient.

$$U_v = \frac{1-2v}{3}(\sigma_{xx}^2 - \sigma_{xx}\sigma_{yy} + \sigma_{yy}^2) \tag{1.7b}$$

$$U_s = \frac{2(1+v)}{3}(\sigma_{xx}^2 - \sigma_{xx}\sigma_{yy} + \sigma_{yy}^2) + 2(1+\nu)(\sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{xz}^2) \tag{1.7c}$$

The strain energy was expressed in terms of equivalent stress $\bar{\sigma}$ as described by Equation 1.8 during the analysis.

$$\bar{\sigma}^2 = 2EU \tag{1.8}$$

where, E = modulus of elasticity, and

U = strain energy.

Failure was assumed to occur at a critical value of U, U_{cr} or at critical value of $\bar{\sigma}$, $\bar{\sigma}_{cr}$. A transient finite element analysis was conducted with Mindlin's plate elements. The effective stresses for the entire analysis was stored. The maximum value of $\bar{\sigma}$ at the Gauss points yielded a distribution of effective stresses. To match the analytical solution with experimental data, two variables: the critical stress $\bar{\sigma}_{cr}$ and the weighting coefficient β, were chosen to fit the experimental data. The same values were used to predict the damaged areas for different cases of impact velocities. It was found that the value of β was usually less than 1, implying that dilatational energy (energy associated with dilatation strain) plays a dominant role in failure mechanism. This failure criterion proved to be adequate for a wide range of impact velocities.

It was shown in many other studies that matrix cracking and delamination were the major modes of damage in low-velocity impact (Chaturvedi & Sierakowski, 1985; Choi & Chang, 1991; Jih & Sun, 1993; Madan, 1991). Wu and Springer (1986) revealed that out-of-plane stress is the major cause of delamination. These authors developed a three-dimensional transient finite element model of a simply supported plate subjected to low-velocity impact. The Tsai-Wu failure criterion was used to predict delamination damage which was shown to agree with experimental data. No material degradation was included in this analysis. Later Wu and Springer (1988) developed a model based on the dimensionless parameter theory relating the delamination dimensions with parameters influencing the size of damage or delamination.

$$l_D = f(\sigma, R, t_f, \Delta Q, D^T, D^B, l_o, K_c) \qquad (1.9)$$

where, σ = is the stress at the location of the damage,

R = is the rate at which the stresses change,

t_f = is the duration of the stresses,

ΔQ = is the difference in reduced stiffness of the two plies adjoining the delamination,

D^T, D^B = are the flexural rigidities of the layers above and below the interface where delamination occurs,

l_0 = initial size of the flaw and

K_c = is the resistance of the material to separation.

The predicted delamination dimensions were within 20% of the experimental results. Again, material degradation during the impact was not considered. This may be the reason for the difference between the analytical results and experimental data.

Choi et al. (1991) conducted experimental studies to show that matrix cracking and delamination play a major role in impact damage. They used a line-nosed impactor to produce a uniformly distributed transient dynamic load across the specimen's width. The following conclusions are drawn:

i) matrix damage was the initial damage due to impact;

ii) delamination follows the "critical" matrix crack;

iii) impact energy threshold is governed by the energy required to initiate the first critical matrix crack;

iv) stacking sequence affects impact resistance of composites.

The experimental studies by Choi et al. (1991), showed that interlaminar shear stresses and in-plane tensile stresses were dominant factors causing matrix cracking. These matrix cracks produce micro-cracks which in turn caused delamination failure. The out-of-plane, stresses on the other hand cause delamination growth. Choi et al. (1991) developed a two-dimensional transient finite element model to verify that matrix damage indeed initiates delamination. A modified Hertzian contact for line-loading was used in conjunction with a matrix failure criterion. The material stiffness of the elements within the damaged area was degraded and the stresses redistributed. Choi and Chang (1992) also developed a three-dimensional transient analysis to verify that out-of-plane stresses cause delamination growth. In this model, in addition to the matrix failure criterion, a delamination failure criterion was also included. To predict damage, the material was first tested for matrix failure and then for delamination, at the locations

where the matrix failed. This model was capable of predicting both matrix cracking and delamination due to the impact.

1.6. Outline of present work

In the present work, another model is proposed to predict damage due to low-velocity impact. The proposed approach is similar to that used by Wu and Springer (1986), except that degradation is included during the impact. Also, instead of the Hertz contact model, an inelastic impact was used which is simpler and effective. To predict the extent of damage, the model utilizes a generally accepted 3D Tsai-Wu failure criterion. While the previous studies did not include material degradation during impact the present model allows for material degradation at regular intervals. The model was implemented using object oriented programming (OOP) concepts to support future extensions. Consequently, the software can be easily modified and specialized for different conditions and applications (e.g., incoporating Hertz's contatct law) without much difficulty.

1.6.1 Organization.

Chapter 2 discusses the mathematical formulations for modeling impact as well as the composite element formulation. Chapter 3 discusses the structure of the software package developed to implement the model. The results and discussions are presented in chapter 4. Conclusions and recommendations are presented in chapter 5.

CHAPTER 2

Formulations

This chapter discusses the mathematical formulation and concepts used to develop the finite element model of impact damage in composite laminates. The approach adopted is similar to the one used by Wu and Springer (1986), with one important exception. The Wu-Springer model did not include material degradation during damage, whereas the present work includes cumulative degradation which can be applied at user-specified intervals, or not applied at all. Section 2.1 provides a brief problem statement. Section 2.2 discusses the various formulations used in the model. Section 2.3 lists the step-by-step analysis procedure.

2.1 Statement of the problem

The objective of the present work is to develop a finite element model to predict damage in composite structures undergoing low velocity impact and accounting for cumulative material degradation during impact. The model is to be validated with a case study involving a rectangular plate whose dimensions are L (length), W (width), and h (thickness), with continuous fibers. The plate consists of n plies whose orientations are arbitrary and need not be symmetric to the midsurface of the plate. An impactor of mass M_p strikes the stationary plate with a velocity V_p as shown in Figure 2.1. While developing the finite element model following conditions were assumed:

i) perfect bonding between plies;

ii) individual layer are homogeneous and orthotropic, and

iii) the impactor adheres to the target after impact.

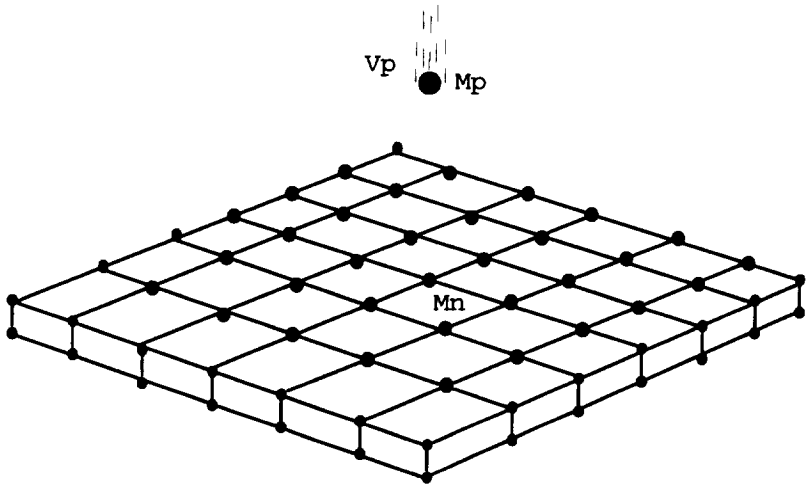


Figure 2.1

Impactor Striking the Rectangular Plate at the Center
(Mp: Mass of the Impactor, Mn: Mass of Impacted Node)

2.2 Analysis

The equation of motion in variational form can be expressed, without damping, as:

$$\int_{\Omega} w_i \rho u_{i,tt} dv + \int_{\Omega} e_{ij} E_{ijkl} \varepsilon_{kl} dv - \int_{\Gamma} w_i \sigma_{ij} n_j dA = 0 \qquad (2.1)$$

where, σ_{ij} = the stress tensor on the boundary of the domain,

ε_{kl} = the strain tensor,

ρ = the density of the plate material,

$u_{i,tt}$ = acceleration vector,

w_i = arbitrary variations,

Ω = the volume of the domain,

Γ = the domain boundary,

n_j = the outward normal vector on the domain boundary,

E_{ijkl} = fourth order tensor represents the material stiffness.

The three terms in Equation 2.1 represent the kinetic energy, the strain energy, and the work done by the boundary forces, in this case the contact forces.

2.2.1 Modeling Contact Behavior. Hertz contact model is a frequently used to predict contact forces. As suggested by Lin and Lee (1989), impact may be modeled as inelastic when the impactor mass is larger than the mass of the impacted node. The authors found that the results obtained from such a model were in good agreement with experimental data. As a result, impact may be modeled as an initial velocity at the impacted node with conservation of momentum taken into account.

A similar inelastic contact model is used in this present work. The conservation of momentum principle dictates that the equivalent initial velocity of the impacted node is given by:

$$V_o = \frac{M_p V_p}{\left(M_p + M_n\right.} \qquad (2.2)$$

where, Mp = the impactor mass,

Vp = the impactor velocity,

Mn = the mass of the impacted node on the plate,

Vo = the equivalent initial velocity of impacted node.

2.2.2 Finite Element Formulation. An 8-noded brick element was used to model the composite plate in a transient finite element formulation. The displacement at any point in the laminate can be expressed as: (Wu & Chang, 1987)

$$u_q = \sum_{r=1}^{8} N_r u_{qr} \quad q = 1,2,3, \tag{2.3}$$

where, u_{qr} are the nodal displacements, and N_r ($r = 1 \sim 8$) is the shape function vector for a 8-noded brick element, N_r can be written as follows:

$$N_1 = (1-\xi)(1-\eta)(1-\zeta)/8,$$

$$N_2 = (1+\xi)(1-\eta)(1-\zeta)/8,$$

$$N_3 = (1-\xi)(1+\eta)(1-\zeta)/8,$$

$$N_4 = (1+\xi)(1+\eta)(1-\zeta)/8,$$

$$N_5 = (1-\xi)(1-\eta)(1+\zeta)/8,$$

$$N_6 = (1+\xi)(1-\eta)(1+\zeta)/8,$$

$$N_7 = (1-\xi)(1+\eta)(1+\zeta)/8,$$

$$N_8 = (1+\xi)(1+\eta)(1+\zeta)/8, \tag{2.4}$$

where ξ, η and ζ are the natural coordinates of the element.

An isoparametric formulation is used such that the coordinates x_q ($q = 1 \sim 3$) of any point inside the element can be expressed in terms of the shape functions:

$$x_q = \sum_{r=1}^{8} N_r x_{qr} \quad q = 1,2,3 \tag{2.5}$$

where x_{qr} are the coordinates of the node r.

The strains at any point in the laminate can be expressed as

$$[\varepsilon]^T = \sum_{r=1}^{8} [B_r] \{u_{1r} \quad u_{2r} \quad u_{3r}\}^T \tag{2.6}$$

where

$$[B_r] = \begin{bmatrix} N_{r,1} & 0 & 0 & 0 & N_{r,3} & N_{r,2} \\ 0 & N_{r,2} & 0 & N_{r,3} & 0 & N_{r,1} \\ 0 & 0 & N_{r,3} & N_{r,2} & N_{r,1} & 0 \end{bmatrix}^T, \quad r = 1 \sim 8,$$

The stresses at any point are given by the following relation

$$\sigma_{ij} = Q_{ijkl}\, \varepsilon_{kl} \quad i,j,k,l = 1\sim3 \tag{2.7a}$$

where Q_{ijkl} is material stiffness tensor, which varies with fiber orientation. The above equation can be reduced to Equation 2.7b by applying material symmetry. Detailed explanation can be obtained from any text book on continuum mechanics or any text book dealing with mechanics of composite materials.

$$\sigma_i = Q_{ij}\, \varepsilon_j \quad i,j = 1\sim6 \tag{2.7b}$$

Q_{ij} for a lamina is computed along its principal material direction i.e., in the direction of fiber, and then Q'_{ij} is obtained by rotating the Q_{ij} by the fiber orientation. The following Equation 2.7c shows the transformations required to rotate Q_{ij} thru angle θ.

$$[Q'] = [T]^{-1}[Q][T]$$

where, [Q'] = material stiffness matrix in the global coordinate system, and

[T] = transformation matrix.

2.2.3 Governing Equations. The finite element equation for transient analysis is given by Bathe & Wilson, 1976.

$$[M]\{\ddot{X}\}+[C]\{\dot{X}\}+[K]\{X\}=\{R\} \tag{2.8}$$

where, [M] = mass matrix,

[K] = stiffness matrix,

[C] = viscous damping matrix (α[M] + β[K]),

α and β are experimentally determined constants,

{X} = nodal displacement vector,

{R} = external load vector.

The basic components involved in the above equation are the mass and stiffness matrices. For computational convenience, a diagonal lumped mass matrix was used. In this method the mass of the element is equally distributed at its nodes. The expression for mass matrix is:

$$\left[M^e\right]=\frac{\rho V^e}{8}[I] \tag{2.9}$$

where, [I] is 24 x 24 identity matrix.

2.2.4 Composite Laminate Element Stiffness Formulation. The formula for stiffness matrix of a finite element is given as (Zienkiewicz, 1977):

$$\left[K^e\right]=\int_{\Omega e}[B]^T[Q][B]dV \tag{2.10}$$

where, [Q] is the material stiffness matrix.

In composite laminates, [Q] varies from layer to layer due to the change in the fiber orientation. There are two methods available to model the composite element. In the first method, each ply is

represented by an element along the thickness of the laminate. This approach results in a large model, making the analysis computationally expensive. In the second approach, each composite element is assumed to contain stacked plies oriented at different angles along the thickness of the element, as shown in Figure 2.2. The second approach allows the material properties to vary in discrete steps through the thickness of the element.

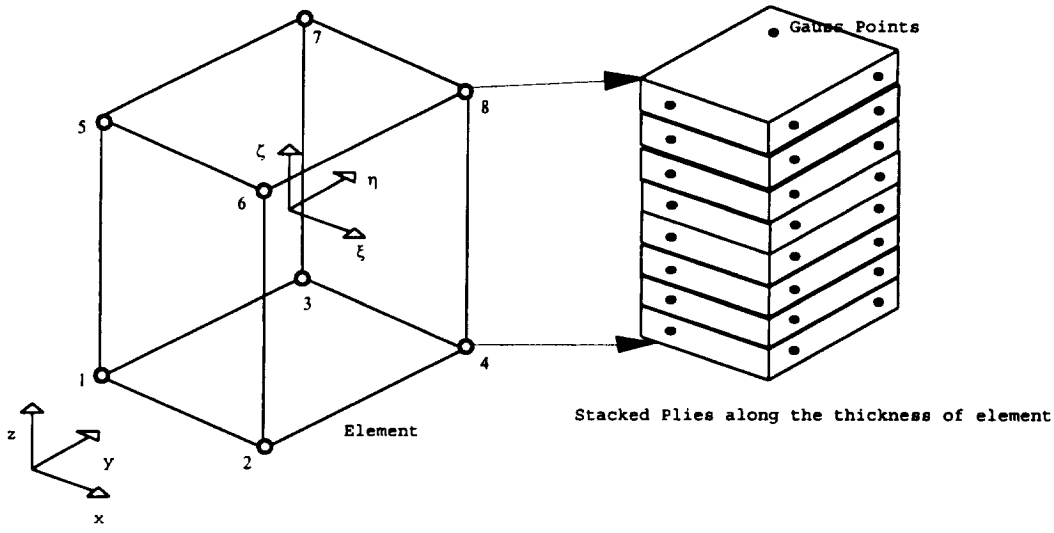


Figure 2.2

Stacked Plies along the Thickness of the Element

The stiffness of a composite element is computed by summing the stiffnesses offered by each ply along the thickness direction of the element:

$$\left[K^e\right] = \iint \left\{ \sum_{k=1}^{k=N} \int_{z_k}^{z_{k+1}} [B]^T [Q]_k [B] dz \right\} dx dy \qquad (2.11)$$

where, N is the number of plies along the thickness of the element, $Z_{k+1}$and Z_k are the top and bottom coordinates of the kth ply in the element. $[Q]_k$ is the material stiffness matrix of the kth ply. The global coordinates x, y, z are mapped into a natural coordinate system having coordinates ξ, η and ζ (Zeinkewicz, 1977).

The stiffness offered by each ply in an element is computed by the summation of the stiffnesses at the Gauss points in the ply. The Gauss points in a ply with 2 X 2 X 1 integration scheme is shown in Figure 2.3.

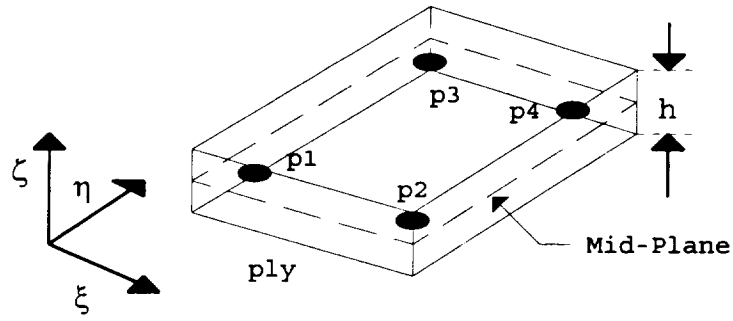


Figure 2.3

Location of Gauss Points for a 2 X 2 X 1 Integration Scheme in a Composite Ply.

2.2.5 Transient Analysis

At any time t, the load vector {R} in Equation 2.8 can be written as a sum of the following components:

$$F_I(t)+F_D(t)+F_E(t)=R(t) \tag{2.12}$$

where, $F_I(t)=[M]\{\ddot{X}\}_t$ are the inertial forces,

$F_D(t)=[C]\{\dot{X}\}_t$ are the damping forces,

$F_E(t)=[K]\{X\}_t$ are the elastic forces.

All of the above variables are time-dependent. Mathematically, Equation 2.8 is a second order differential equation which is solved using direct integration. In the direct integration method, the derivatives are approximated by finite differences and integrations may be carried out implicitly or explicitly.

In the implicit method, the equation of motion is satisfied at the next time step, i.e., at t+Δt. The following equation illustrates the Newmark method:

$$[M]\{\ddot{X}\}_{t+\Delta t}+[C]\{\dot{X}\}_{t+\Delta t}+[K]\{X\}_{t+\Delta t}=\{R\}_{t+\Delta t} \tag{2.13}$$

$$\{\dot{X}\}_{t+\Delta t}=\{\dot{X}\}_t+(1-\gamma)\Delta t\{\ddot{X}\}_t+\gamma\Delta t\{\ddot{X}\}_{t+\Delta t} \tag{2.14}$$

$$\{X\}_{t+\Delta t}=\{X\}_t+\Delta t\{\dot{X}\}_t+(0.5-\beta)\Delta t^2\{\ddot{X}\}_t+\beta\,\Delta t^2\{\ddot{X}\}_{t+\Delta t} \tag{2.15}$$

where, β and γ are the experimentally determined parameters.

Using the above Newmark finite difference Equations 2.14, and 2.15 for displacement and velocity vectors at t+Δt, the terms $\{\ddot{X}\}_{t+\Delta t}$ and $\{\dot{X}\}_{t+\Delta t}$ can be derived in terms of $\{X\}_{t+\Delta t}$. These expressions, when substituted in the Equation 2.13 above, result in the following equation expressed in terms of displacement vector $\{X\}_{t+\Delta t}$ at time t+Δt:

$$[\hat{K}]\{X\}_{t+\Delta t} = \{\hat{R}\}_{t+\Delta t} \tag{2.16a}$$

where, $[\hat{K}]$ is the effective stiffness matrix defined as:

$$\hat{K} = K + a_o M + a_1 C \tag{2.16b}$$

$$a_o = \frac{1}{\alpha\,\Delta t^2}; \quad a_1 = \frac{\delta}{\alpha \Delta t} \qquad \delta \geq 0.5; \quad \alpha \geq 0.25(0.5+\delta)^2,$$

and

$\{\hat{R}\}_{t+\Delta t}$, is the effective load vector defined by:

$$\{\hat{R}\}_{t+\Delta t} = \{R\}_{t+\Delta t} + [M](a_o X_t + a_2 \dot{X}_t + a_3 \ddot{X}_t) + [C](a_1 X_t + a_4 \dot{X}_t + a_5 \ddot{X}_t), \tag{2.16c}$$

$$a_2 = \frac{1}{\alpha \Delta t}; \quad a_3 = \frac{1}{2\alpha} - 1,$$

$$a_4 = \frac{\delta}{\alpha} - 1; \quad a_5 = \frac{\Delta t}{2}\left(\frac{\delta}{\alpha} - 2\right).$$

The solution of Equation 2.16a is substituted back in the finite difference equations to compute the velocity and acceleration vectors at time t+Δt. There are many other finite difference methods available, namely the Houbolt method, the Wilson θ method, and Central difference method. The procedure involved in all these methods is similar (Bathe & Wilson, 1976). The advantage of implicit method is that large time steps can be used. The disadvantages of implicit methods are (Belytschko, 1983):

i) The methods are computationally expensive since they involve the assemblage and inversion of $[\hat{K}]$. Also they are not suitable for structural problems involving progressive

failure because of the numerical difficulties that often result from material degradation (Wolcott & Yener, 1989);

ii) They demand greater storage space because of the complexity and size of software;

iii) They cannot be applied easily to nonlinear problems as the changes that must be made to the software make the methods even more computationally expensive.

2.2.6 Explicit Integration Method. In this method the equation of motion is satisfied at the current time step t, i.e.,

$$[M]\{\ddot{X}\}_t+[C]\{\dot{X}\}_t+[K]\{X\}_t=\{R\}_t \tag{2.17a}$$

$$\{\ddot{X}\}_t=[M]^{-1}\{\{R\}_t-[C]\{\dot{X}\}_t-[K]\{X\}_t \tag{2.17b}$$

As inelastic impact and no damping were assumed the equation 2.17b may be expressed as:

$$\{\ddot{X}\}_t=[M]^{-1}\{-[K]\{X\}_t \tag{2.17c}$$

Using the present acceleration vector, the velocity and displacement vectors can be derived for the t+Δt time step using simple finite difference equations:

$$\{\dot{X}\}_{t+\Delta t}=\{\dot{X}\}_t+\{\ddot{X}\}_t*\Delta t \tag{2.18}$$

$$\{X\}_{t+\Delta t}=\{X\}_t+\{\dot{X}\}_t*\Delta t \tag{2.19}$$

The displacement and velocity vectors computed at time step t+Δt are used to update internal resisting forces. This procedure is carried out for the entire impact time duration to obtain the complete response of the system.

Following are the advantages and disadvantages of the explicit integration methods (Belytschko 1983):

i) Fewer computations are required in each time step;

ii) The algorithm is simple in logic and structure;

iii) They requires little core space compared to implicit methods;

iv) They are ideal for testing new ideas, because it requires less coding;

v) They are conditionally stable, so that very small time steps may be required.

The small time steps are compensated for by simple computation required in each time step. To assure stability, the time step must always be less than a critical time step computed from the largest eigen-value of the system. Critical time is computed using (Belytschko, 1983):

$$\Delta t_{cr} = 2/\sqrt{\lambda_{max}} \qquad (2.20)$$

where Δt_{cr} is critical time step and λ_{max} is the maximum eigen value computed on an element-by-element basis. The explicit method was adopted for present work.

2.3. Modeling of Damage.

Failure in composites is complicated by a multitude of interacting mechanisms including fiber breakage, micro-buckling, delamination, fiber pullouts, and matrix cracking. The Tsai-Wu failure criterion (Tsai, 1971) is used to determine overall damage without distinguishing different types of failure. The failure criterion for orthotropic material is expressed by the following inequality:

$$F_1\sigma_x + F_2\sigma_y + F_3\sigma_z + F_4\sigma_{yz} + F_5\sigma_{zx} + F_6\sigma_{xy}$$
$$+ F_{11}\sigma_x^2 + 2F_{12}\sigma_x\sigma_y + 2F_{13}\sigma_x\sigma_z + F_{22}\sigma_y^2 + 2F_{23}\sigma_y\sigma_z + F_{33}\sigma_z^2$$
$$+ F_{44}\sigma_{yz}^2 + F_{55}\sigma_{zx}^2 + F_{66}\sigma_{xy}^2 \geq 1 \qquad (2.21)$$

where, σ_x, σ_y, σ_z are stresses along X, Y, Z axes, respectively and σ_{yz}, σ_{zx}, σ_{xy}, are shear stress in Y-Z, Z-X and X-Y planes, respectively.

The coefficients F_i, F_{ij} are given as:

$$F_1 = \frac{1}{X} - \frac{1}{X'}, \quad F_{11} = \frac{1}{XX'},$$

$$F_2 = \frac{1}{Y} - \frac{1}{Y'}, \quad F_{22} = \frac{1}{YY'},$$

$$F_3 = \frac{1}{Z} - \frac{1}{Z'}, \quad F_{33} = \frac{1}{ZZ'},$$

$$F_4 = \frac{1}{Q} - \frac{1}{Q'}, \quad F_{44} = \frac{1}{QQ'},$$

$$F_5 = \frac{1}{R} - \frac{1}{R'}, \quad F_{55} = \frac{1}{RR'},$$

$$F_6 = \frac{1}{S} - \frac{1}{S'}, \quad F_{66} = \frac{1}{SS'},$$

$$F_{12} = \frac{1}{2P^2}\left[1 - P\left(\frac{1}{X} - \frac{1}{X'} + \frac{1}{Y} - \frac{1}{Y'}\right) - P^2\left(\frac{1}{XX'} + \frac{1}{YY'}\right)\right],$$

$$F_{13}=\frac{1}{2P^2}\left[1-P\left(\frac{1}{X}-\frac{1}{X'}+\frac{1}{Z}-\frac{1}{Z'}\right)-P^2\left(\frac{1}{XX'}+\frac{1}{ZZ'}\right)\right],$$

$$F_{23}=\frac{1}{2P^2}\left[1-P\left(\frac{1}{Y}-\frac{1}{Y'}+\frac{1}{Z}-\frac{1}{Z'}\right)-P^2\left(\frac{1}{YY'}+\frac{1}{ZZ'}\right)\right],$$

where, X and X′ are tensile and compressive strengths along the fiber or X, direction,

Y and Y′ are tensile and compressive strengths of the material along transverse, or Y, direction,

Z and Z′ are tensile and compressive strengths of the material along normal, or Z, direction,

Q and Q′ are positive and negative pure shear strengths on the Y-Z plane; R and R′, on the Z-X plane; and S and S′ along X-Y plane, and

P is experimentally determined by application of biaxial tension.

2.3.1 Material Degradation. Material degradation can be either be complete or partial. In the complete degradation method, the stiffness at failed Gauss point is completely removed from the overall element stiffness:

$$[K'] = [K]_{prev} - [K]_{failed\ points} \qquad (2.22)$$

where $[K']$ is damaged stiffness matrix,

$[K]_{prev}$ is the previous stiffness matrix,

$[K]_{failed\ points}$ the stiffness at the failed points.

In the partial degradation method, only a fraction of the stiffness at the failed Gauss points is removed:

$$[K'] = [K]_{prev} - \Phi * [K]_{failed\ points} \quad (2.23)$$

where Φ is the degradation factor ($0 \leq \Phi \leq 1$).

2.4 Analysis Procedure

The following steps summarize the analysis procedure:

i) A 3-D finite element model is developed.

ii) Transient analysis of the finite element model is carried out to determine the impact response.

iii) During the analysis, nodal displacements and internal resisting forces are computed for every time step Δt. These displacements are used to compute stresses and strains at the Gauss points. Also, strain energy, kinetic energy, and total energy are computed, and recorded for the whole model.

iv) The computed stresses at the Gauss points are checked with the Tsai-Wu criterion for material integrity.

v) The Gauss points at which the material fails are recorded, i.e., the Gauss point coordinates. Depending on the material degradation method, the element stiffness may be either completely or partially degraded.

vi) After material degradation, the acceleration vector at current time step is computed. Using the accelerations, the displacements, velocities and the internal resisting forces at step $t+\Delta t$ are determined.

vii) Steps iii thru vi are repeated for the duration of interest.

viii) The failed Gauss point locations recorded during the analysis are used to plot the damage regions.

CHAPTER 3

Implementation

The software package developed is based on an object-oriented-implementation framework called Extensible Implementation Framework for Finite Elements (EIFFE). Section 3.1 discusses the fundamentals of Object Oriented Programming (OOP) concepts. Sections 3.2 and 3.3 describe the various EIFFE classes as well as classes developed to implement the model described in chapter 2.

3.1 OOP Concepts

Early engineering software was written in procedural languages that modeled logical units as black boxes. Every unit of code is boxed off so that it remains concealed. Such boxes are called functions in C and procedures in Pascal (Eckel, 1993). The major drawback of procedural programming is that it lacks control over data. The designers of procedural programming languages designed the languages based on the assumption that the code required no maintenance or extension. Such assumptions worked when the project size was small, but failed miserably when the complexity of the problem increased.

To accommodate the complexity of large programs, the concept of structured programming was introduced. In structured programming, large programs are divided into modules. Each module is further divided into sets of related procedures that manipulate data. Although structured programming improved clarity, reliability, and ease of maintenance of software, large-scale programs continues to pose challenges, due to the following reasons:

i) Structured programming require a great deal of planning to generate software that is extensible, maintainable and bug-free.

ii) The software developed is usually rigid and intractable; modification is difficult due to the interaction of code from different modules.

In, the 1980s, a new programming paradigm called OOP evolved, which alleviated some problems. While procedural programming hides the complexity of operations performed on data, OOP hides both the data and the operations (Eckel, 1993). An object-oriented language emphasizes data types and the intrinsic operations that may be performed on those data types. Data do not flow openly around a system as in procedural programs, but are protected from accidental modification. In OOP, function calls are replaced by message passing. Messages cause objects to manipulate data.

3.1.1 OOP Terminology. The following are some terms often used in object oriented languages.

Object: An Object is an instance of a class, is defined by a set of attributes, (e.g., a geometrical object would have points, color, and size as the variables) and a set of procedures (rotate, move, expand, contract) that operate on these attributes.

Class: A class is a template for creating objects which share common attributes, and methods. A class not only defines the object's attributes but also provides methods for manipulating these attributes. The following example defines a class called CPerson:

```
class CPerson
{
private:
      CDate m_DateofBirth;
      float m_Age;
      char* m_Name, m_SSNumber, m_Address, m_Telephoneno;
public:
      void Person(char* Name, char* SSNumber, CDate DateofBirth);
      void EnterAddress(char* Address);
      void EnterTelephoneNumber(char* TelephoneNo);
      float GetAge();
      char * GetAddress();
      char * GetTelephoneNo();
      virtual void DisplayData();
      virtual void PrintData();
};
class CDate
{
private:
      int month, day, year;
public:
      SetDate(int, int, int);
      GetDate(int *, int *, int *);
};
```

Data Encapsulation: The isolation of data from the external environment is called data encapsulation. This feature is also known as data hiding. Since the data stored in the object is inaccessible, their modification can be performed only through a controlled interface. Data Encapsulation promotes modularity. In the above example, data such as name, social security number, address in class CPerson cannot be accessed directly. However, they can be manipulated using CPerson's interface, i.e., by calling its methods.

Inheritance: Inheritance defines a relationship among classes, wherein a class inherits the attributes and behavior of one or more other classes. This feature allows the software to be reusable. The following class CStudent example illustrates the principle of inheritance and the software reusability feature of OOP.

```
class CStudent : public CPerson
{
private:
  char* m_University, *m_ClassNo;
  float m_GPA;
public:
  CStudent(char* Name,char* SSNumber,char* DtofBirth,char*Univ,
  char* ClassNo): CPerson(Name, SSNumber, DtofBirth){};
  char * GetUniversity();
  char * GetClassNo();
  float GetGPA();
  void DisplayData(COutput OutputObject);
};
```

Attributes such as name, social security number, and date of birth are inherited from class CPerson. The relationship can be clearly understood if student is seen as a kind of person with some additional attributes. Other attributes, such as the name of university, and classes the student is attending, are added to the CStudent class. As most of the attributes and methods are inherited, little work is needed to create the class CStudent.

Polymorphism: It is a concept wherein different objects may respond differently to the same message depending on the classes the objects belong to. A message "DisplayData" in the above example, sent to a computer console object results in data printed on the computer monitor, whereas the same message sent to printer object results in data being printed on paper. The message is the same but the response is different.

3.1.2 Advantages of OOP. OOP provides better security and reusability than procedural programming in many ways:

i) Objects are self-contained entities that can be introduced into a system without much difficulty, since any bug associated with the new code is localized to the object itself.

ii) New object types may be derived from previously defined ones. This saves time and supports quick explorations. New objects do not modify the behavior of parent objects and keep new bugs localized in the new objects.

iii) Well designed classes support code reusability and extensibility and lead to greater productivity.

iv) Partitioning of work comes naturally, thus making delegation of work in large projects easier.

v) Data encapsulation promotes secure systems.

vii) Software maintenance and management becomes much easier due to encapsulation and uniform object interfaces.

viii) Data Encapsulation increases readability and reduces the need for documentation.

3.2 EIFFE Class Libraries

EIFFE consists of five class libraries:

i) Matrix classes,

ii) Function classes,

iii) Finite element classes,

iv) Material classes,

v) Finite element collection classes.

The first level organization is shown in Figure 3.1. Various types of matrix objects such as symmetric, banded symmetric, rectangular matrices, and vectors may be created using the matrix class library. Basic matrix operations such as matrix addition, subtraction, multiplication, and several other specialized techniques for solving matrix equations are also defined. Different types of functions such as monomial, bivariate, trivariate, and polynomial function can be created and evaluated at any point with the function objects created from the function class library. The finite element class library defines various finite element objects such as nodes, boundary conditions, and elements that are required to build a finite element model. The type of material the model uses can be defined as anisotropic, orthotropic, transversely isotropic, or isotropic material objects defined in Material class library of EIFFE. The methods for computing different kinds of stiffness matrices such as plane-stress, and plane-strain stiffness matrices are also defined. Material failure computations using the Tsai-Hill criterion and Tsai-Wu criteria are included. Finite element objects can be stored in collection objects, such as element list, node list, and function list. The collection classes are included

in the finite element collection class library. A brief description of finite element classes used to develop the present software is described.

3.3 Class Descriptions

3.3.1. CObject The CObject class sits at the base of the Microsoft Foundation Class (MFC) Library. The CObject class contains common methods, such as serialization, run-time class information, and object diagnostic output. The EIFFE class libraries use CObject as the base class as shown below:

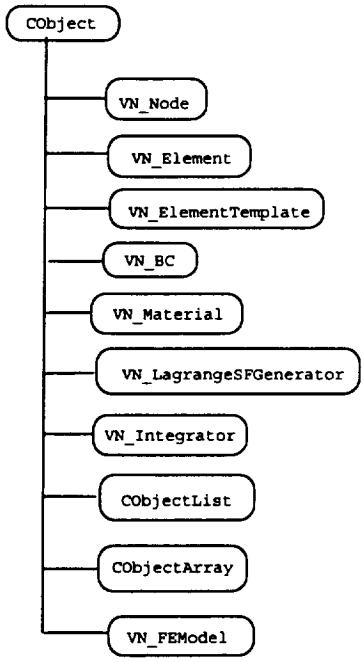


Figure 3.1

The Top Level Class Hierarchy of EIFFE

3.3.2. VN_ElementTemplate The VN_ElementTemplate class has methods defined to compute shape functions, shape function gradients, and the Jacobian matrix. These methods are used in the computation of strain-displacement matrices, stiffness matrices, internal or initial force vectors, and element stress and strain vectors. Since different element types may perform some of these computations differently, these methods may need to be implemented or overridden in the derived classes that represent particular finite element formulations. The template object is invoked by an element object to perform certain computational tasks (computation of element stiffness matrix, initial force vector, etc). Before other types of element template classes are explained, it is important to understand the relationship and communication process between an element object and element template object.

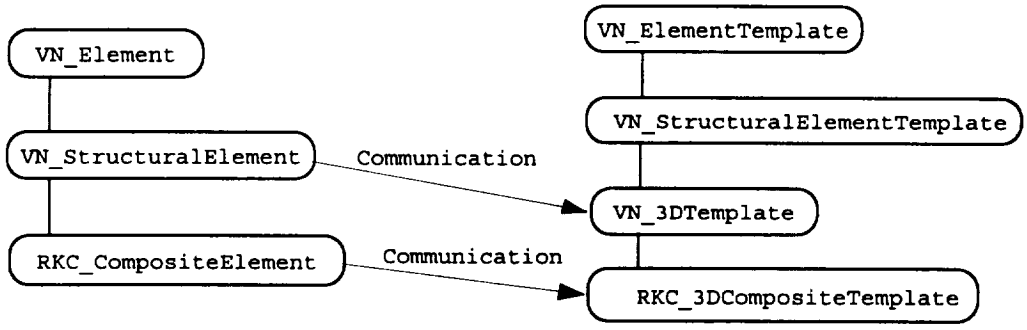


Figure 3.2

Class Hierarchy of Element Template and Element classes

3.3.3 Communication between Element and Element Template Objects

When an element object is asked to compute its matrices, i.e., stiffness matrix, initial force vector or mass vector, it forwards the message to the attached element template object. The template object, before processing any request, checks whether the requesting element is of the right kind. The computational methods vary with the type of element (2D or 3D), and the element order, that is, the order of shape functions (linear, quadratic, etc). The template object also checks whether the requesting element has the right number of nodes to ensure compatibility between shape functions specified for the template and the number of nodes in the element, e.g., Four nodes are required for 2D element with linear shape functions, nine nodes are required if shape functions are quadratic. The template object uses the nodal information provided by the element object to compute the Jacobian matrix. It then requests the material object to compute the material stiffness matrix which is required in computation of the element stiffness matrix and the initial force vector. Once the element template object has all the required components, it creates the appropriate function integrands and requests the suitable integrators to carry out integrations. The computed matrix or vector is finally returned to the requesting element. Figure 3.2 shows the communication process between various element objects and element template objects.

3.3.4. VN_Element: The VN_Element class defines element objects. It includes the basic methods for computing stiffness matrices and stress and strain vectors. These methods simply call the corresponding methods in the template classes. Each element object also encapsulates

the connectivity data needed to evaluate the Jacobian matrix. The element object assembles its matrices, and vectors, into the global matrices and vectors. Similarly, it retrieves element matrices and vectors from global matrices and vectors. The hierarchies of element classes and element template classes are shown in Figure 3.2.

3.3.5. VN_StructuralTemplate: VN_StructuralTemplate is a subclass of VN_ElementTemplate that deals with structural analysis. The methods defined deal with computation of matrix and vector objects, such as, stiffness matrix, body force vector, initial force vector, mass matrix, stress and strain vectors, etc. The structural element template object uses suitable integrators to integrate different types of function integrand objects provided to it. Using different element formulations, function integrand objects are created by VN_3DTemplate, VN_PlaneStrainTemplate, and VN_PlaneStressTemplate, which are subclasses of VN_StructuralTemplate class.

3.3.6. VN_3DTemplate: This class is a subclass of VN_StructuralTemplate. All the computational methods required by a 3-dimensional element are defined in this class. This class is capable to compute the strain-displacement matrix [B], for a 3-dimensional elements. It uses the [B] matrix and material stiffness matrix, [Q], to create the suitable 3D integrand functions such as stiffness integrand function, initial force integrand function, etc. The order of the functions generated depends upon the shape function generator used when creating the template object.

3.3.7. VN_StructuralElement: The VN_StructuralElement class inherits from the VN_Element class. VN_StructuralElement objects are used in finite element models dealing with structural problems. Structural problems can be solved using any one of the finite element formulations such as plane strain element, plane stress element, or 3D brick element. A structural element object can thus use any of the template objects created from VN_PlaneStressTemplate, VN_PlaneStrainTemplate or VN_3DTemplate. Structural element objects delegate the task of computing their matrices and vectors to the appropriate structural template object. The template object performs the computations and returns the results to the structural element object.

3.3.8. RKC_CompositeElement: The RKC_CompositeElement inherits from VN_StructuralElement. A composite element object is assumed to consist of multiple plies stacked along the thickness direction. The composite element object keeps track of the thickness and the lay-up sequences of the plies. This information is passed to the composite template object upon request. The node and ply numbering scheme used, are shown in Figure 3.3. The composite element also tracks the damage status at the Gauss points. When the composite element is asked to update its stiffness, it in turn requests the composite template object to compute the stiffness degradation at the damaged points. The element object either completely or partially removes the stiffness at the damaged points depending on the degradation method chosen.

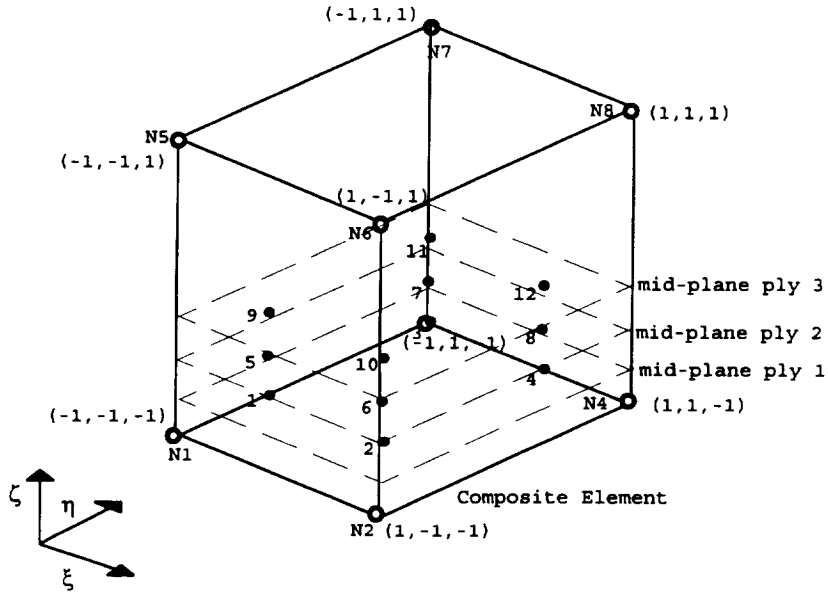


Figure 3.3

Node Numbering and Ply Numbering Scheme in a Composite Element

3.3.9. RKC_3DCompositeTemplate: This class is similar to the VN_3DTemplate class, with difference being that this class uses an orthotropic or transversely isotropic material object to model the composite material properties. The composite template object requests the material object to computes its material stiffness in the fiber coordinate system and then transforms it to the global coordinate system. It also has methods to compute stresses and strains in various plies. Methods are defined in this class to check for failure in the material using the Tsai-Wu criterion. After checking for failure it updates the material status in the element object by keep track of failed points.

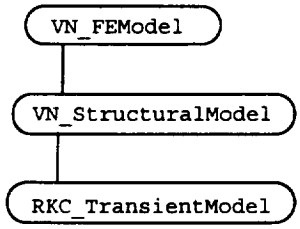


Figure 3.4

The Class Hierarchy of Finite Element Model Classes

3.3.10. VN_FEModel: VN_FEModel is the manager of the whole finite element model. VN_FEModel stores and keeps track of all the elements, nodes, and boundary condition objects belonging to the finite element model. Finite Element related objects can be added or removed. VN_FEModel has a GO method that instructs the model to proceed with the analysis. The hierarchy of finite element model classes is shown in Figure 3.4.

3.3.11. VN_StructuralModel: This class is a subclass of VN_FEModel that handles structural analysis models. The VN_Structural Model object processes user request by sending its own message to the objects it contains in the correct order. For example, after the creation of a finite element mesh, each element object in the model is requested by the structural model object to its own element stiffness matrix. When the structural model is requested to solve the model, it sends messages to element objects to request the assembly of their element matrices and vectors. After assembly, the model object issues a message to the appropriate solver object to obtain the analysis results.

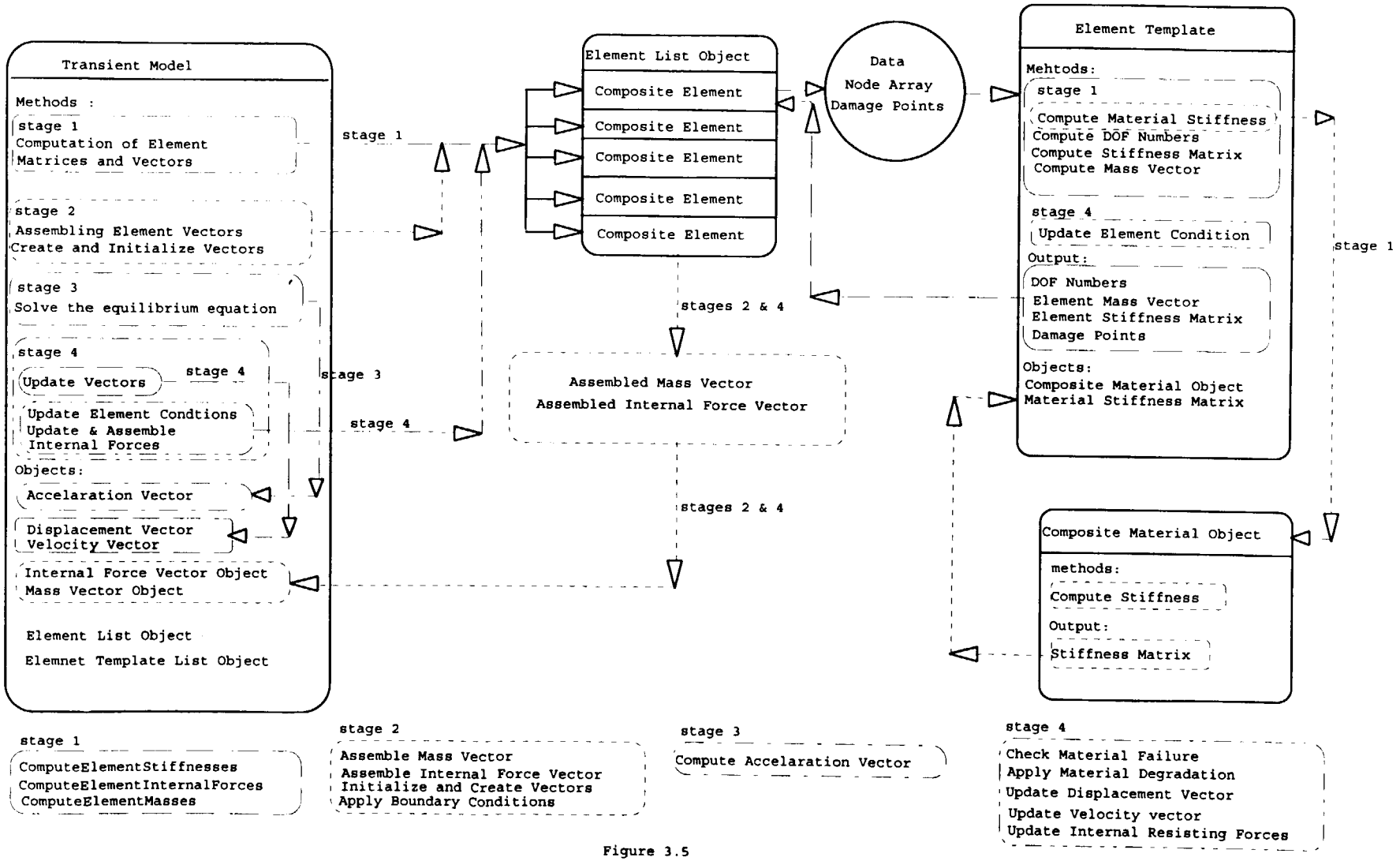


Figure 3.5

Communication Model of Transient Finite Element Model

3.3.12. RKC_TransientModel: This class inherits from VN_StructuralModel. The transient model object is used for solving transient problems wherein the solution is sought at different time marks. The methods involved could be classified into four stages, the different stages are shown in Figure 3.5.

Stage 1: The transient model object calls all the elements to compute their degrees of freedom, mass vectors, and stiffness matrices. The element objects in turn requests its element template object to compute the element vectors and matrices. The template object send message to material object to compute the material stiffness matrix which is used in the computation of stiffness matrix. The computed element matrices and vectors are stored in element object.

Stage 2: The transient model object creates empty vectors for nodal displacements, velocities, accelerations and internal resisting forces. It also sends message to each element object to assemble its element vectors into global vectors. Both the initial conditions and the essential boundary conditions are also applied.

Stage 3: The equations of motion are solved for nodal accelerations.

Stage 4: The velocity and displacement vectors are updated for the next time step. The elements are requested to update their state, i.e., check and record any material failure as well as updating the stiffness matrix. Internal resisting forces are recomputed using updated stiffness matrix and displacement vector. Stages 3 and 4 are repeated for the entire duration.

CHAPTER 4

Case Studies

Four case studies were used to validate the model and to illustrate the analytical capabilities built into the model. In section 4.1, the impact response of a simply supported steel beam was studied. The composite element formulation is verified in section 4.2 by solving the transient response of a steel plate and a composite plate, both subjected to a suddenly applied uniform pulse load. The inelastic impact response of a composite plate is verified in section 4.3 by solving a numerical example taken from Lin and Lee (1990). Finally, the ability to predict damage in a composite laminate is demonstrated in section 4.4.

4.1 Impact Response of a Steel Beam

The impact response of a simply supported steel beam was predicted using both plane strain and brick elements to validate the transient solver. Beam, as shown in Figure 4.1, has dimensions of 10 x 1 x 1 in. A spherical steel impactor with equal mass to that of the beam, impacts the beam at its center with a velocity of 10 in/sec. The impactor is assumed to adhere to the beam during the impact. The following represents the material properties used for both impactor and the beam:

Material : Steel,

Density : 0.285 lbs/ in^3,

Poissons Ratio υ : 0.29,

Youngs Modulus : 30×10^6 lbs/in^2.

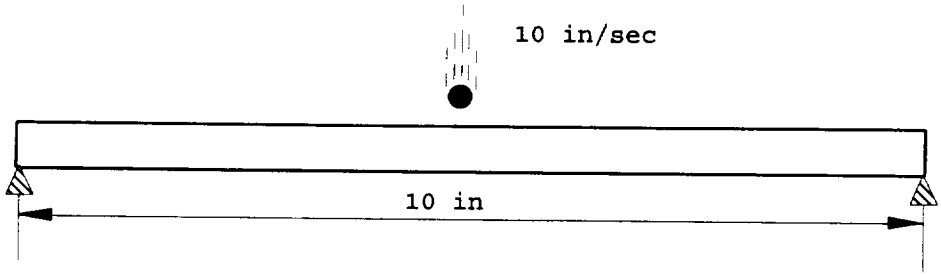


Figure 4.1

Simply Supported Steel Beam.

As the model is symmetric, only half of the beam was modeled using both plane-strain and brick elements. A 2 x 2 Gaussian quadrature formula was used for plane strain elements and a 2 x 2 x 2 quadrature formula was used for the brick elements. The impact response of the model was obtained for an impact duration of 1000 μs.

The deflection at the center of the beam is shown in Figure 4.2. The displacement values are about 5% lower than those obtained by Goldsmith (1960), who replaced the beam with an equivalent spring-mass system. Since his formulation did not account for shear deformation in the beam, all impact energy was imparted to flexural mode greater deflections that should be observed in the actual impact. In the present case, both brick and plane strain elements include shear; hence some impact energy is channeled to the shear deformation mode, resulting in smaller deflections than that obtained by Goldsmith.

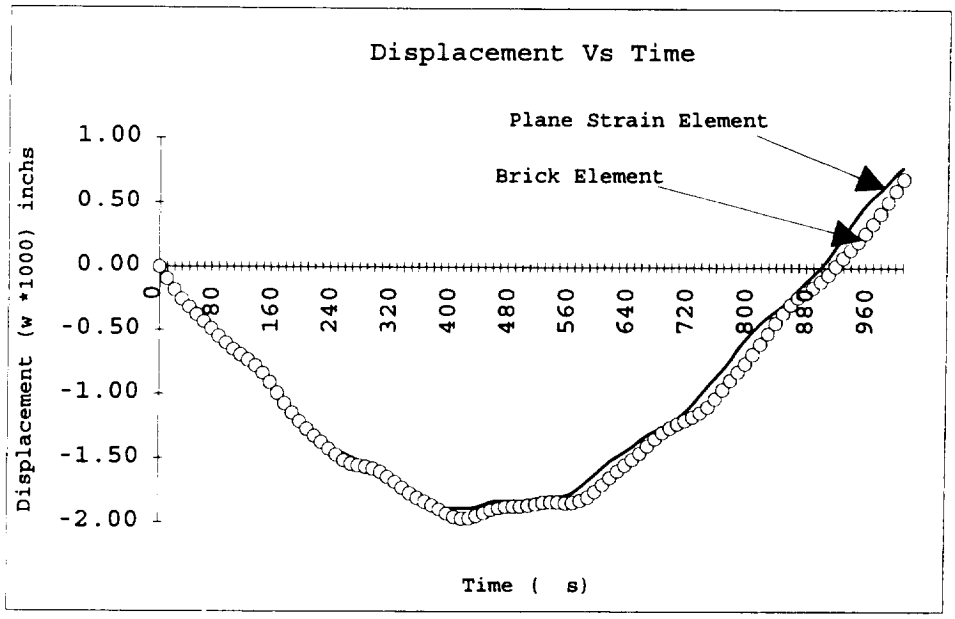


Figure 4.2

Deflection at the Center of the Simply Supported Beam.

Figure 4.2 also shows that the plane strain model yields smaller deflections than does the brick model. The reason for this discrepancy is that the plane strain assumption imposes an additional constraint that results in stiffer elements than the brick counterparts.

A simple approximate analytical solution may be derived when the impactor mass is large compared to the mass of the beam. In such cases, the system is represented by a spring-mass system in which the mass of the beam constitutes little to the overall response and can be represented as one half the total beam mass lumped at the center. An equivalent spring with the stiffness equal to the flexural stiffness of the beam is used. The impactor mass is also lumped at the beam center. A half period for this model computed to be 955 μs whereas the finite

element model predicts 910 μs, with a difference of 5%. It should be recognized that spring-mass model yields an upper limit as no energy absorption due to shear deformation is assumed. On the other hand the finite element should form a lower limit due to the constraints imposed by the assumed shape functions. Higher-order elements should improve the accuracy of the finite element solution. Thus, the actual deflection and time period should not be further than 5% from the finite element solution.

4.2 Transient Response of a Plate.

To validate the composite element formulations, the transient responses of two types of plates was predicted. The first plate was made of an isotropic material. The second plate was made of a transversely isotropic laminated material.

4.2.1 Isotropic Plate

A square plate, simply-supported, subjected to uniform pulse load is used to further validate the model. Due to symmetry, only a quarter-plate was modeled. The finite-element model, plate dimensions, and material properties used are shown in Figure 4.3, as follows:

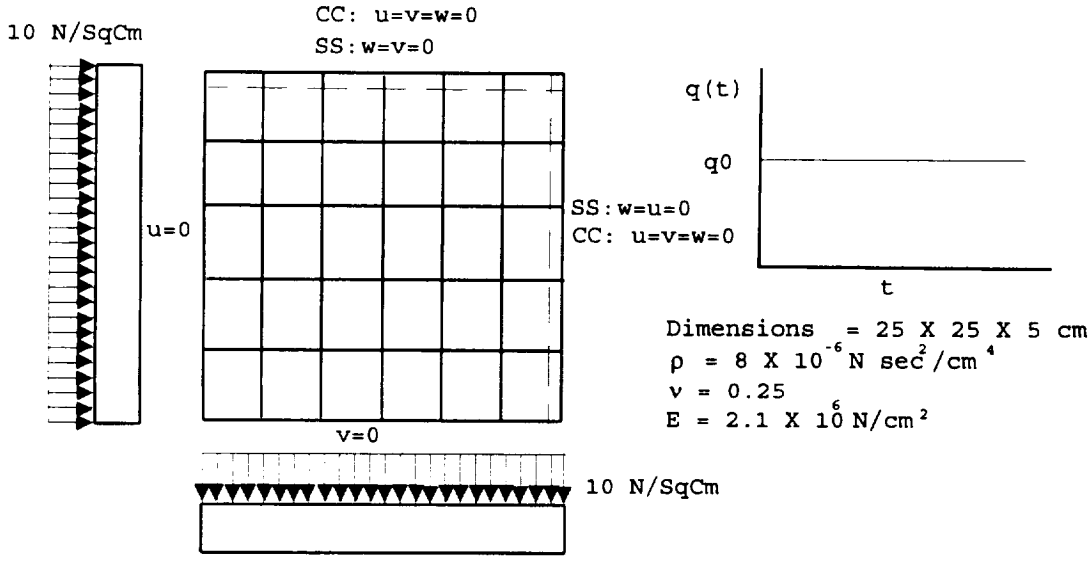


Figure 4.3

Finite Element Model of a Quarter Plate.
(SS: Simply Supported; CC: Clamped)

A finite element model using brick elements with a mesh density of 5 x 5 x 2 was used to model the quarter plate. The numerical computations were carried out using a 3 x 3 x 3 Gaussian quadrature formula. Figure 4.4, shows the deflection at the center of the plate. The deflection of the center obtained was very close to the result obtained by Reddy (1983), even though the element formulations used were different. Figure 4.5, shows the variation of strain, kinetic energy, and the total system energy with time.

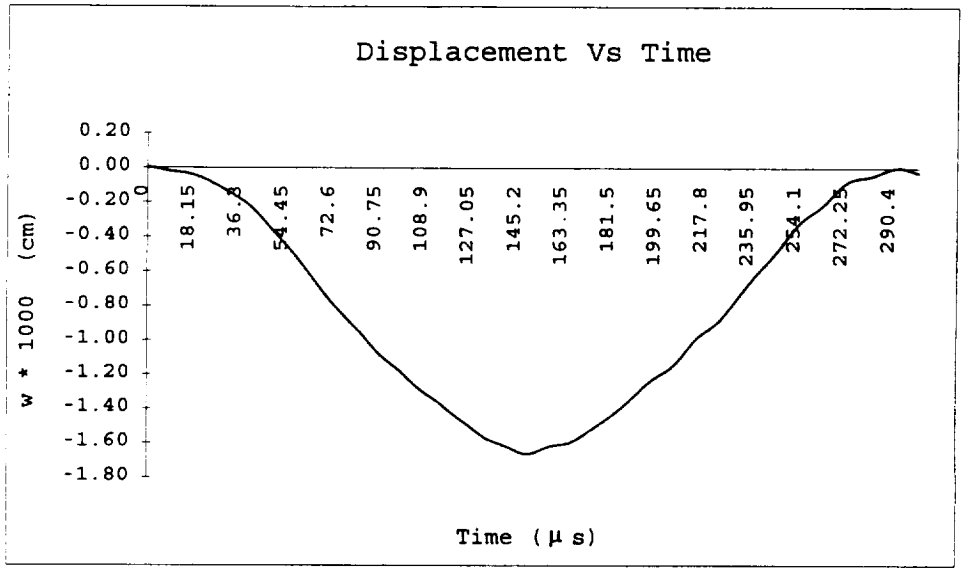


Figure 4.4

Transient Response at the Center of a Simply Supported Square Isotropic Plate subjected to Suddely Applied Uniform Pulse Load.

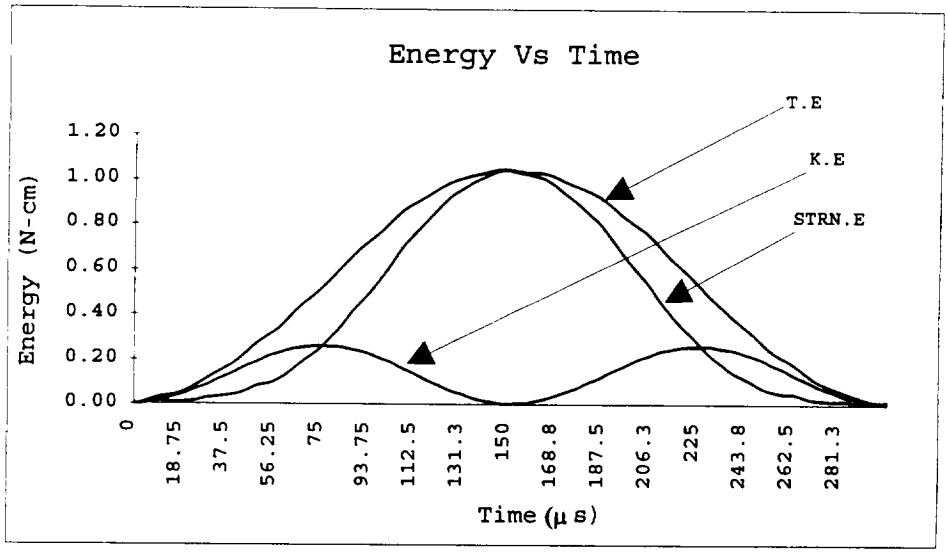


Figure 4.5

Variation of Energies in a Square Isotropic Plate subjected to Suddenly Applied Uniform Pulse Load. (T.E: Total System Energy; K.E: Total System Kinetic Energy; STRN.E: Total System Strain Energy)

When the load is applied, the kinetic energy increases as the nodal velocities increase. Simultaneously, the strain energy increases due to the resistance of the plate to the external force. The same resistance slows the nodal movements, causing a decrease in kinetic energy. Eventually, the strain energy becomes maximum when the kinetic energy vanishes. Afterwards, the nodes reverse direction and a mirror image is obtained.

4.2.2 Composite Plate

The previous plate model was repeated using a composite material with $E_1 = 25E_2$ and $G_{12} = 0.5E_2$ and $E_2 = 2.1 \times 10^6$ N/cm^2. Different lay-up sequences and two boundary conditions (clamped or simply supported) were tested:

(i) [-45/45] lay-up sequence: A finite element model with mesh density 5 x 5 x 2 was used to model a composite laminate with a [-45/45] lay-up sequence. Each ply along the thickness was represented by an element with the respective fiber orientation. A 3 x 3 x 3 Gaussian quadrature formula was used. Figure 4.6 as follows, shows the deflection at center of the plate for both boundary conditions:

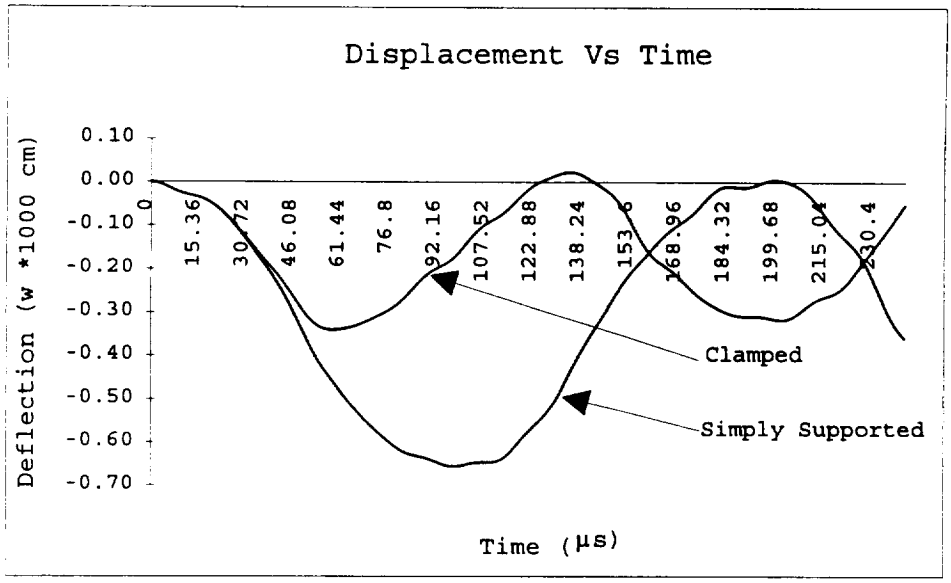


Figure 4.6

Deflection at the Center of a Square Composite Plate with a Lay-Up Sequence [-45/45] subjected to Suddenly Applied Pressure Loading.

(ii) [30/45/90/0] Lay-up Sequence: A mesh density of 3 x 3 x 1 was used to model a laminate with [30/45/90/0] lay-up sequence. Also only one element was used along the laminate thickness direction. Figure 4.7, shows the center deflection of a clamped plate.

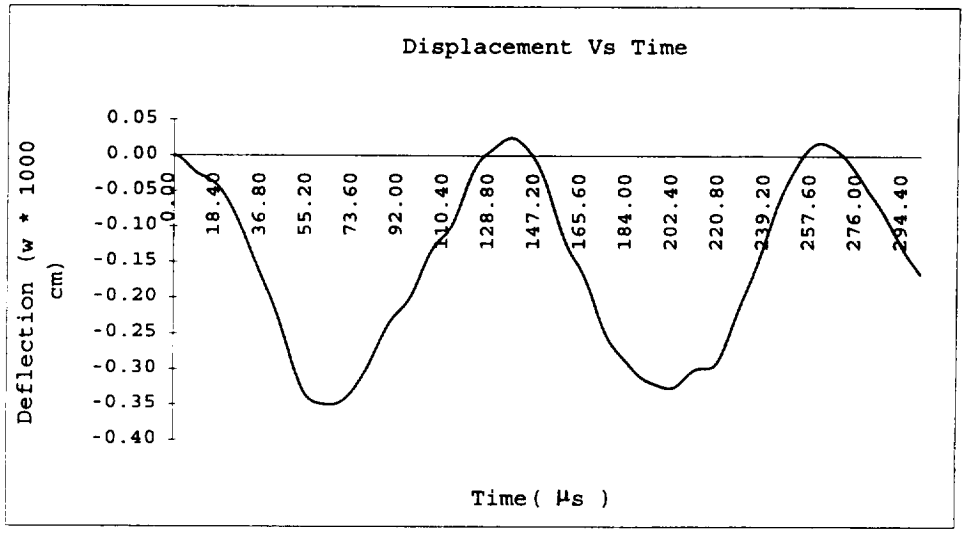


Figure 4.7

Deflection at the Center of a Square Composite Plate with Lay-Up Sequence [30/45/90/0] subjected to Suddenly Applied Uniform Pulse Load.

The results obtained from present software were close, and were comparable to those published by Reddy(1983). The maximum difference observed was 6%. This variation was due to differences in element formulation, and the time integration schemes employed. These examples sufficiently demonstrate the accuracy of the present model.

4.3 Inelastic Impact Response of a Composite plate

In the present work, inelastic impact was assumed because of its simplicity. However, Hertz contact law can be incorporated with some minor modifications to the software. A numerical case study from Lin and Lee (1990) was used to validate inelastic impact response of a composite plate. A clamped composite plate having dimensions 0.14 m x 0.14 m with a lay-up sequence of $[0_5/90_5/0_5]$ and the following material properties are used:

E_1 = 40 Gpa,

E_2 = 8.27 Gpa,

$G_{12} = G_{13} = G_{23}$ = 4.14 Gpa,

ν_{12} = 0.26,

h = 0.00335 Mt.,

ρ = 1901.5 kg/m^3.

The impactor has a mass of 0.014175 kg and an initial velocity of 39.7 m/s. Due to symmetry, only a quarter plate was modeled using a mesh density of 10 x 10 x 1. 2 x 2 x 1, and 2 x 2 x 2 integration schemes were used for computation of stiffness matrices of plies and element mass matrices, respectively. The displacement response, and the energy response, are shown in Figures 4.8 and 4.9, respectively.

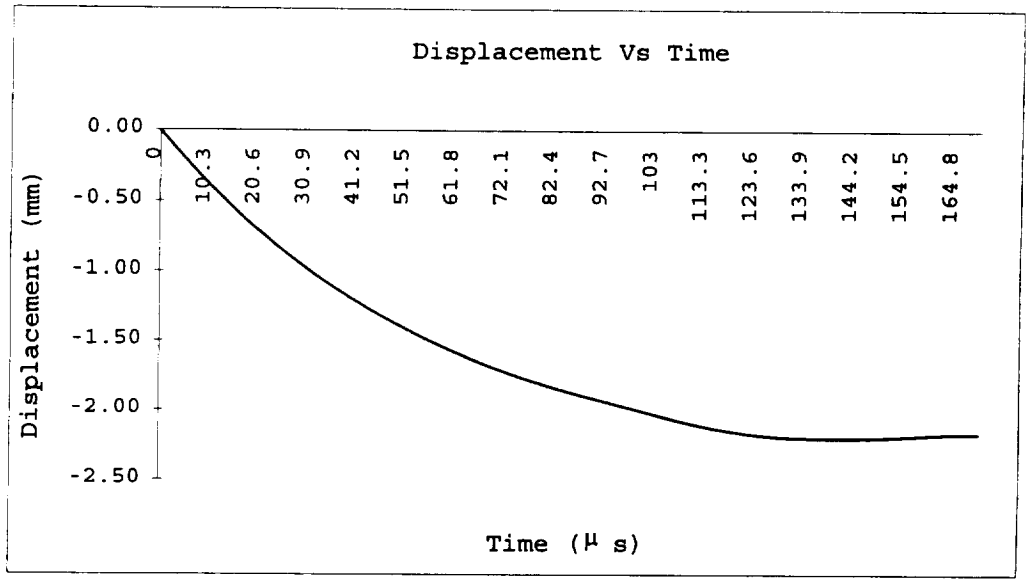


Figure 4.8

Deflection Response at the Center of a Clamped Composite Plate due to Inelastic Impact.

Both the maximum deflection, and the time period, are lower than the results obtained by Lin and Lee (1990) by 10-15%. This difference can be attributed to the fact that the shell element formulation used by Lin and Lee does not include shear deformation. A better result may be obtained by using higher order approximation functions that culminate in a computationally larger model.

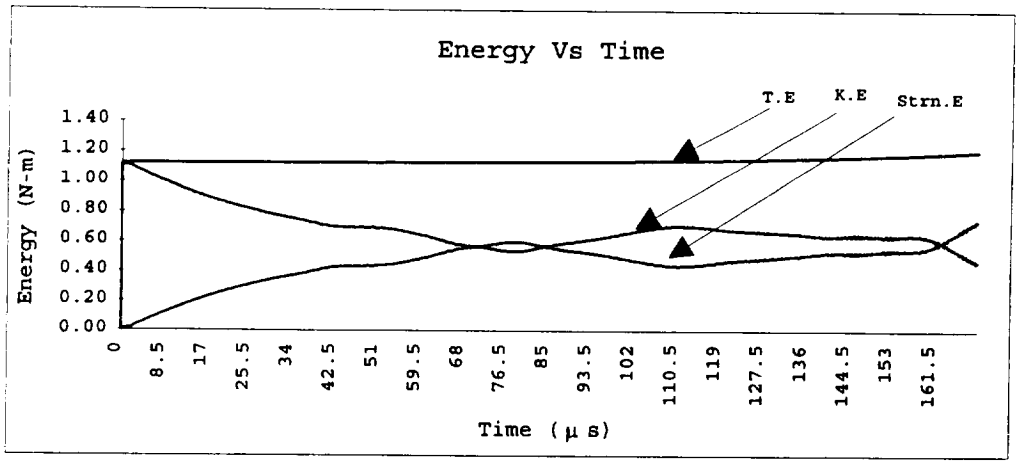


Figure 4.9

Variation of Energies in a Clamped Composite Plate due to Inelastic Impact. (Strn.E: Strain Energy; K.E: Kinetic Energy; T.E: Total Energy)

It is clear from Figure 4.9 that the total system energy is slowly increasing, though it is expected to remain constant (with no damping). This increase in energy is due to numerical inaccuracies that are carried over from previous iterations. Accumulated numerical inaccuracies can cause the system to become unstable. The onset of instabilities can be observed in the strain energy and kinetic energy curves at the 142 μs mark. This unstable behavior can be avoided in two ways with some trade-offs:

i) Use smaller time steps. However, a very small time step could result in computationally expensive analysis

ii) Choose appropriate damping parameters to correct for the energy creepage as well as to account for structural damping.

4.4 Damage Prediction in a Composite Plate

A model from Choi and Chang (1992) was analyzed in two case studies with the following conditions:

i) Material degradation is not included in the analysis.

ii) Material degradation is included in the analysis. Either complete or partial degradation could be applied. Complete degradation was assumed in this case study.

4.4.1 No Material Degradation

A failure analysis of a model with no material degradation was solved. The problem consists of a spherical steel impactor moving with a velocity of 7.8 m/sec and stationary composite plate. The plate made of Fiberite T300/976 having the dimensions 10 cm x 7.6 cm x 0.403 cm and a lay-up sequence of $[0_4/-45_4/45_4/90_4/45_4/-45_4/0_4]$. The properties of the composite material are shown in Table 4.1. The impactor hits the plate at the center and is assumed to stick to the plate after impact. A finite element model of a quarter plate using a 5 x 4 x 1 size mesh was used. The model is shown in Figure 4.10. Stiffness computations were carried out using a 3 x 3 x 1 integration scheme while a 3 x 3 x 3 integration scheme was used for computation of a diagonal mass matrix. A time step 1e-09 sec interval was used in this analysis.

Ply thickness, h (mm)	0.144
Density, ρ (Kg/m^3)	1540
Longitudinal Young's Modulus, E_{xx} (GPa)	156
Transverse Young's Modulus, E_{yy} (GPa)	9.09
Shear Modulus in x-y direction, G_{xy} (GPa)	6.96
Poisson's ratio in x-y direction, υ_{xy}	0.3
Poisson's ratio in y-z direction, υ_{yz}	0.3
Longitudinal tensile strength, S_{LT} (MPa)	1520
Longitudinal compressive strength, S_{LC} (MPa)	1590
Transverse tensile strength, S_{TT} (MPa)	45
Transverse compressive strength, S_{TC} (MPa)	252
Longitudinal shear strength, S^{o} (MPa)	105

Table 4.1

Material Properties of Fiberite T300/976 Graphite/Epoxy
source : (Choi and Chang 1992).

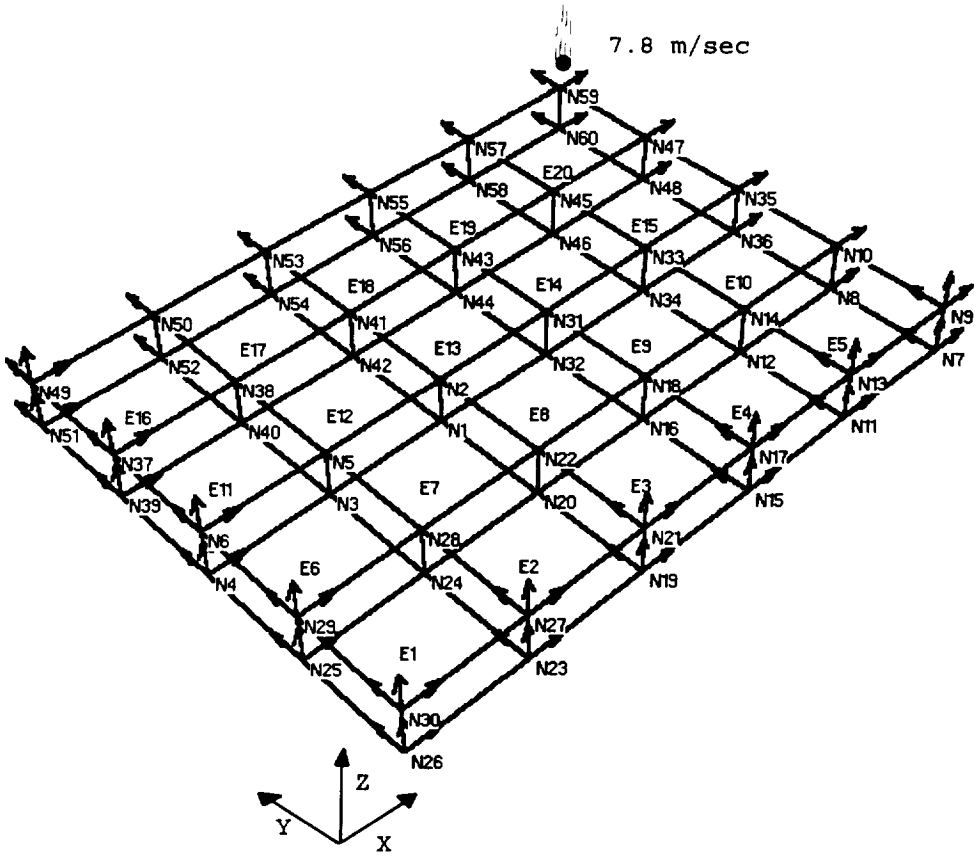


Figure 4.10

Quarter Plate Finite Element Model of a Clamped Composite Plate with Lay-Up Sequence of $[0_4/-45_4/45_4/90_4/45_4/-45_4/0_4]$.

Figures 4.11 and 4.12 show the impact response of the impacted node and the force profile of the internal resisting force at the impacted node.

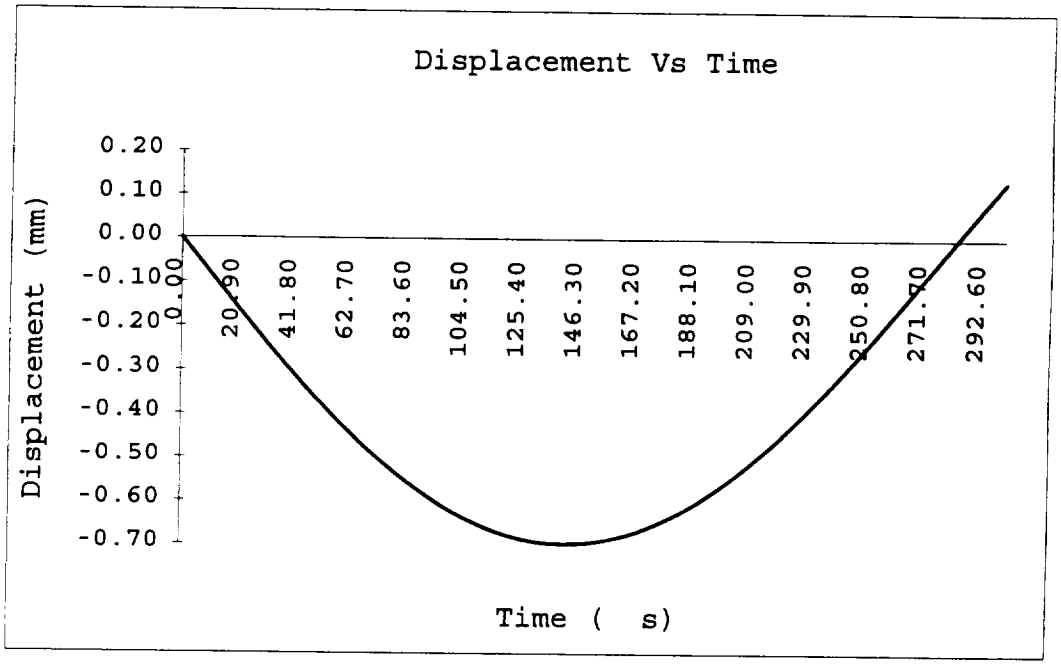


Figure 4.11

Deflection Response at the Center of the Composite Plate Model with No Material Degradation.

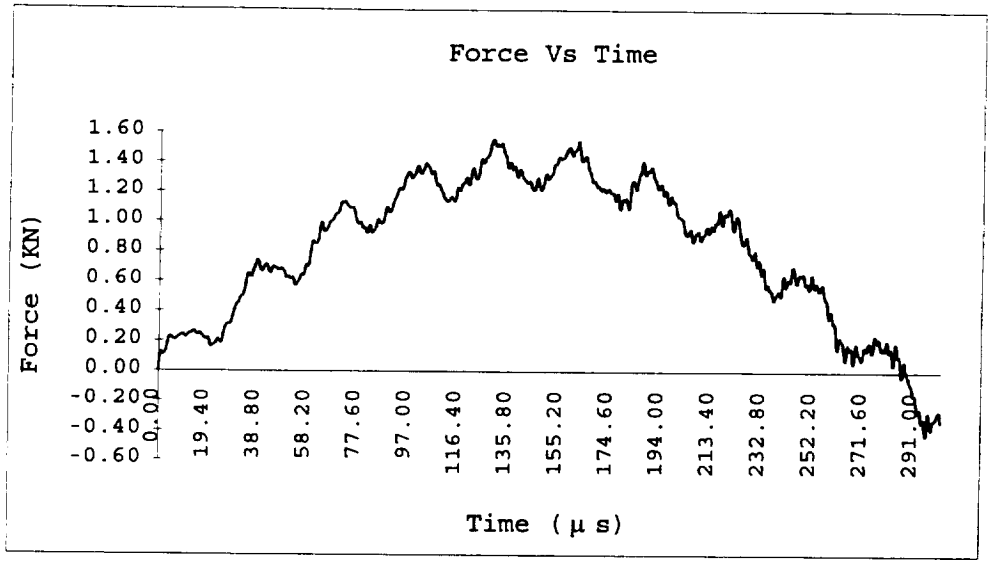


Figure 4.12

Resisting Force Response of the Impact Point in Composite Plate Model with No Material Degradation

The impact response obtained could not be verified as impact response for this particular example was not studied by Choi and Chang (1992). However, the resisting force and deflection profiles obtained at the center of the plate due to impact were similar.

It is a well-known fact that damage in a system cannot occur without energy dissipation. The variation of energy with time is an important parameter to predict and to estimate damage. In the present case, the material damage was not included in the analysis, hence the total energy remained constant as expected as shown in Figure 4.13.

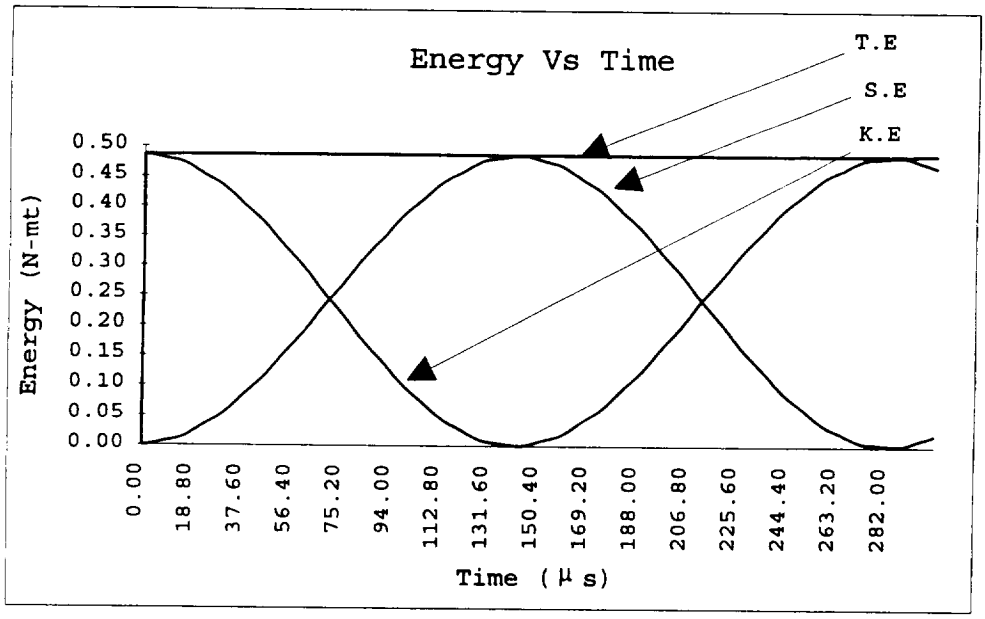


Figure 4.13

Variation of Energies in a Composite Plate Model with No Material Degradation.

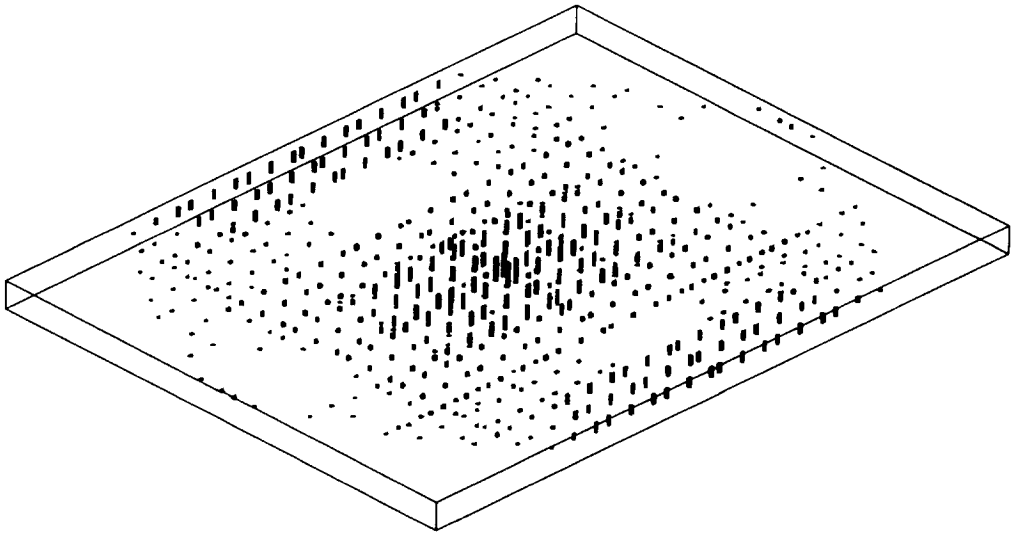
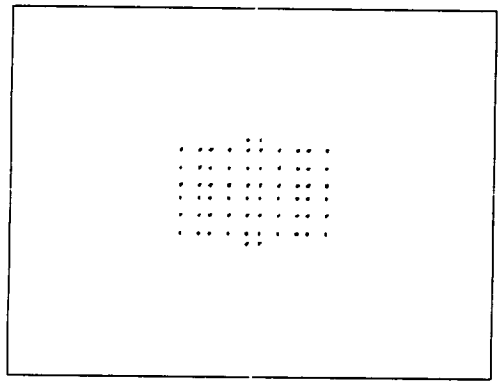


Figure 4.14

Isometric View of the Predicted Damage with Model using No Material Degradation Method.

Figure 4.14 illustrates the predicted damage, i.e., the damage zone. Each dot represents a Gauss point at which material degradation is expected to occur.

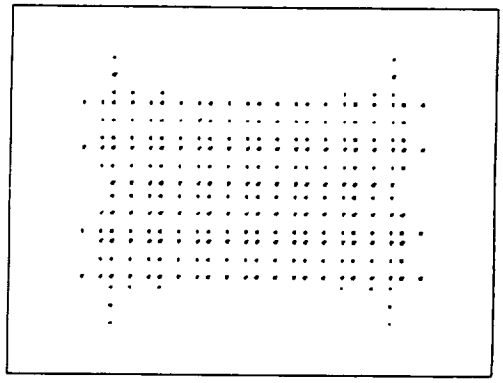
Figures 4.15(a-g) shows damage in each layer as well as the overall damage.



Layer 1

$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

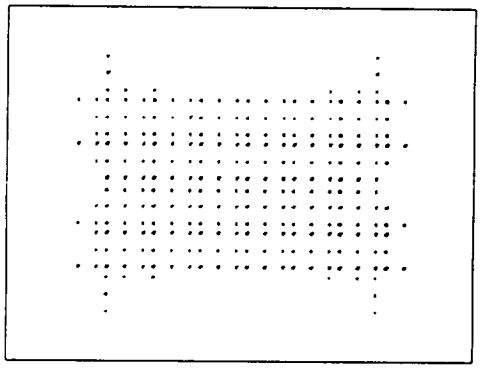
(a)



Layer 2

$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

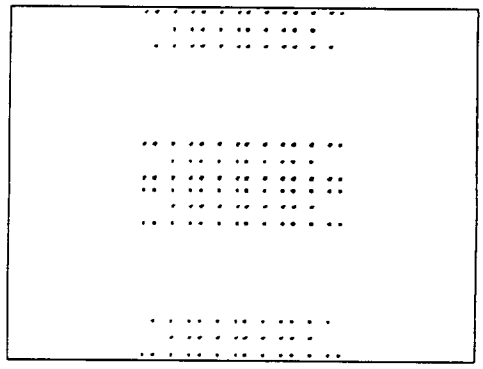
(b)



Layer 3

$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

(c)



Layer 4

$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

(d)

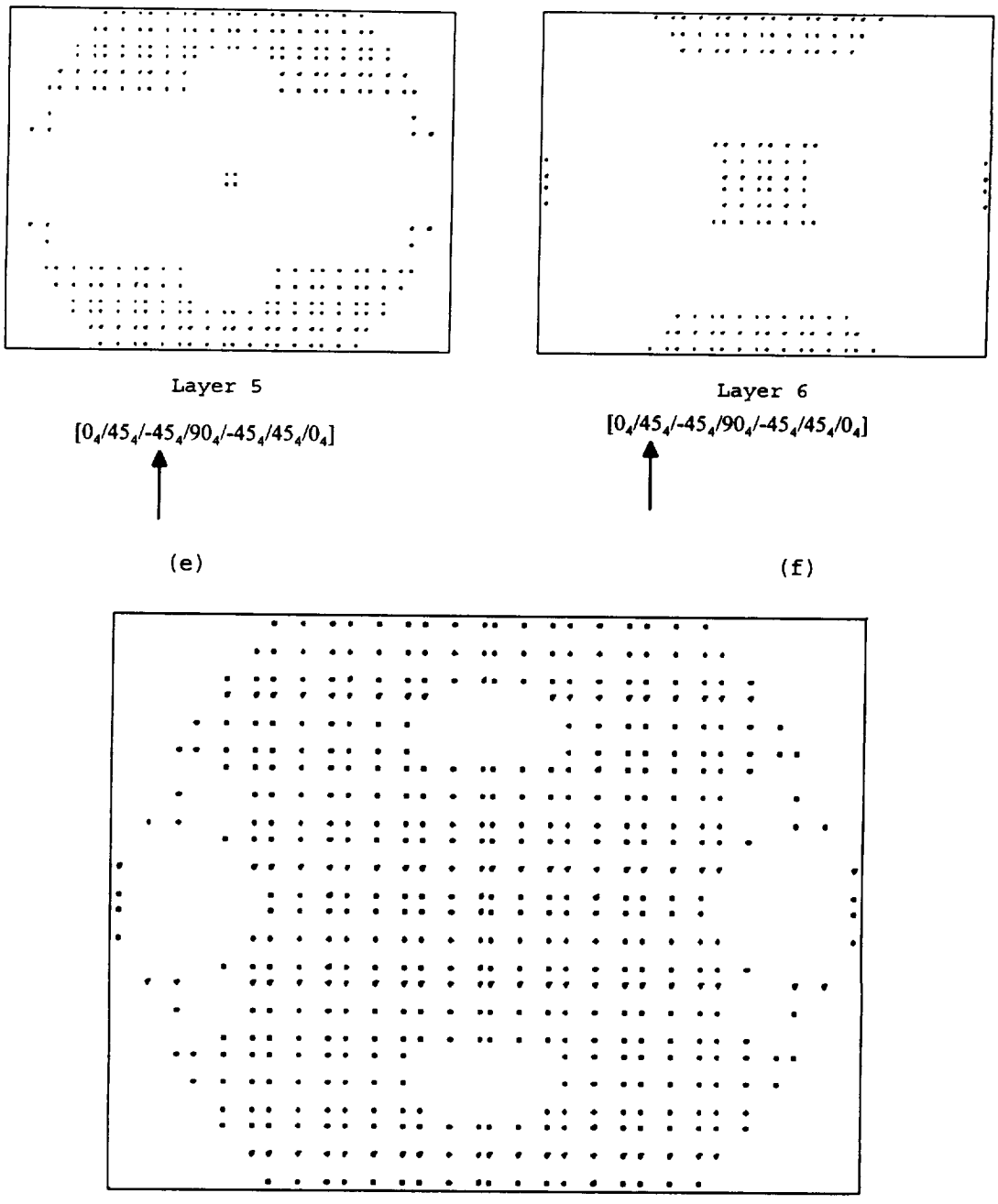


Figure 4.15

Top View of the Predicted Damage with Model using
No Material Degradation Method.
Top View of Damage in (a) Layer 1 (b) Layer 2 (c) Layer 3
(d) Layer 4 (e) Layer 5 (f) Layer 6 (g) All Layers

4.4.2 Material Degradation

In the material degradation model the stiffness of the model has to be degraded, should failure occur. There are two methods available to degrade material after failure. The material may be completely degraded or it may be partially degraded. In the present case complete degradation of material was assumed. Figure 4.16 shows the deflection of the impacted node. The larger time period and deflection indicate that the material has softened during impact.

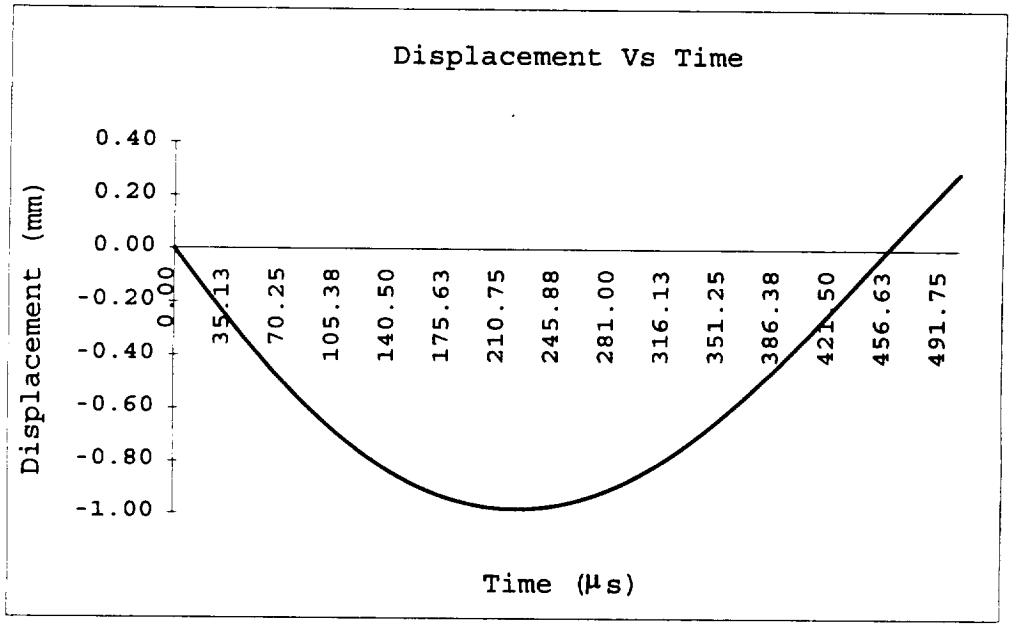


Figure 4.16

Deflection Response at the Center of a Composite Plate Model with Material Degradation.

Figure 4.17, shows the resisting force at impacted node. As expected, resisting force is smaller than that obtained in the no degradation model. The irregularity in the force profile clearly indicates when damage occurs.

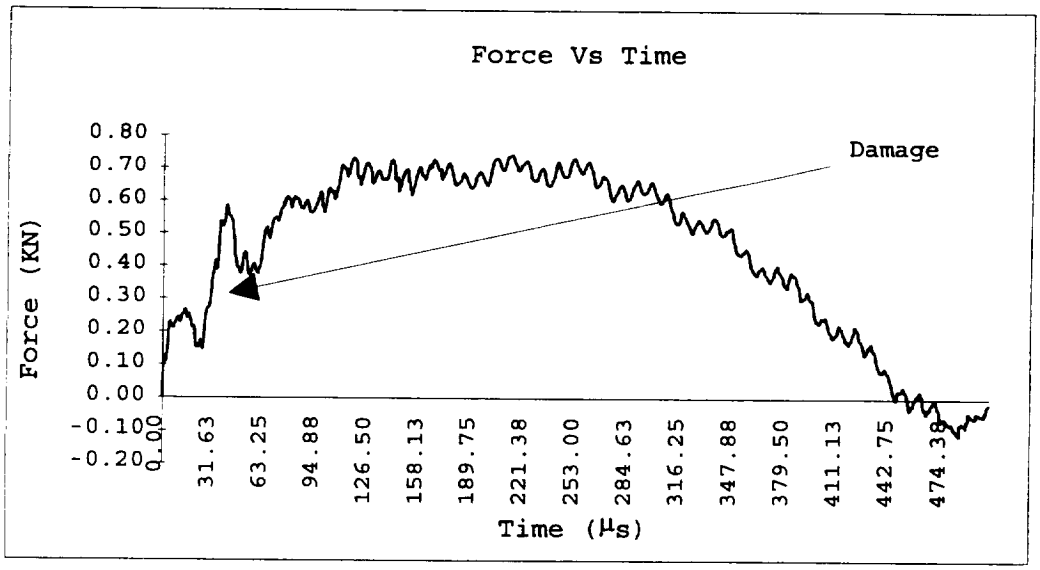


Figure 4.17

Resisting Force Response of the Impact Point in a Composite Plate Model with Material Degradation

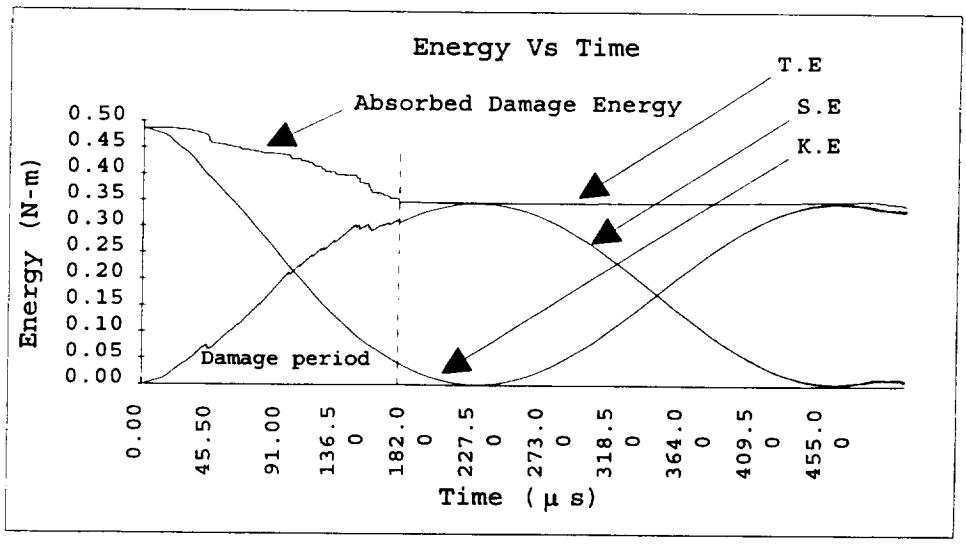


Figure 4.18

Variation of Constituent Energies in a Composite Plate Model with Material Degradation.

Energy dissipation caused by damage is illustrated in the Figure 4.18. The total energy curve indicates a gradual drop between 0 and 100 μs, after which no further damage was incurred except at the very end. As there is no other source that consumes energy, damage is cause of the observed energy dissipation. Using Figure 4.18, the extent of damage may be estimated by the severity of the energy loss. Once the threshold value for energy loss is determined experimentally, the actual residual strength may be predicted from these curves.

Figure 4.19, shows the predicted damaged area, and also, shows the extent of damage in plies through the thickness of the laminate. While Figures 4.20 (a-h) shows damage in each layer and the overall damage.

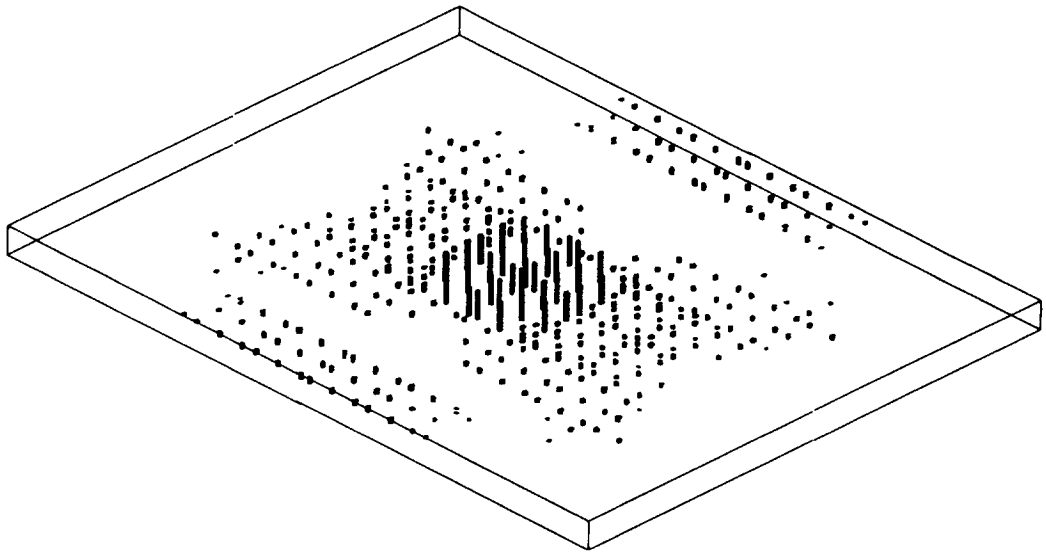
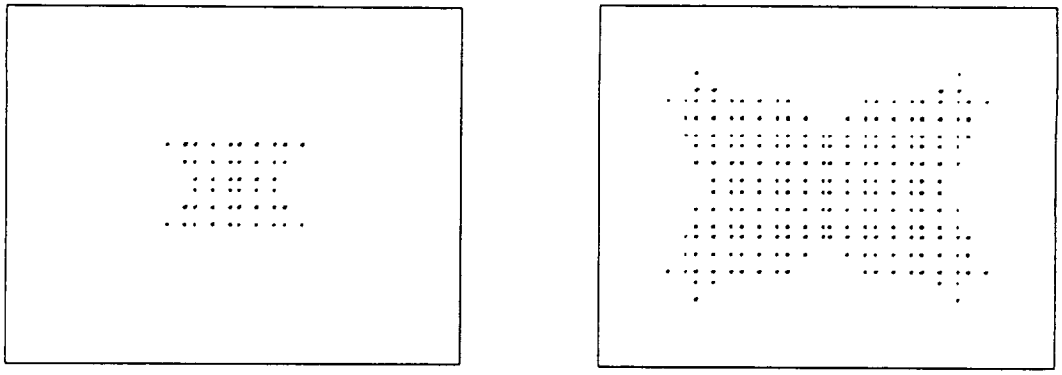


Figure 4.19

Isometric View of the Predicted Damage with Model using Material Degradation Method.



Layer 1

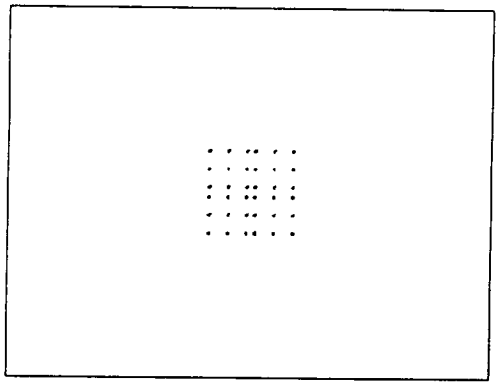
$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

(a)

Layer 2

$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

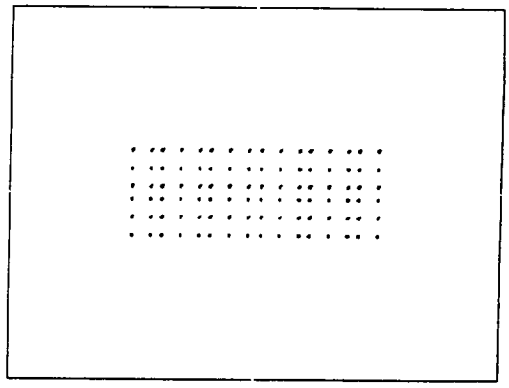
(b)



Layer 3

$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

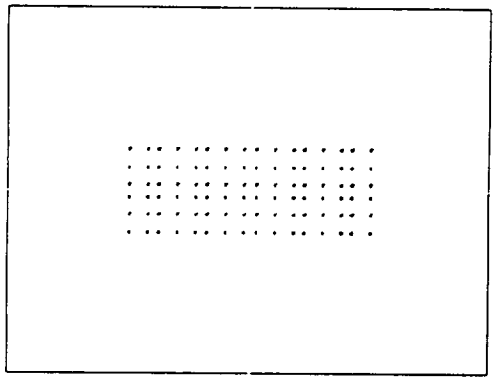
(c)



Layer 4

$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

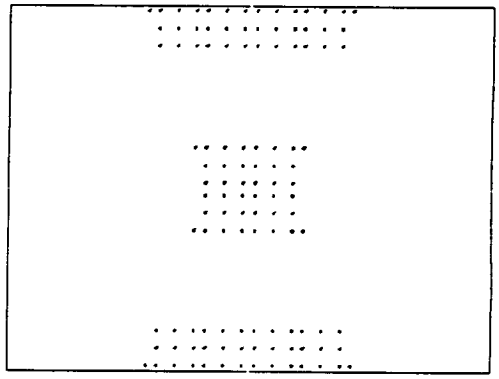
(d)



Layer 5

$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

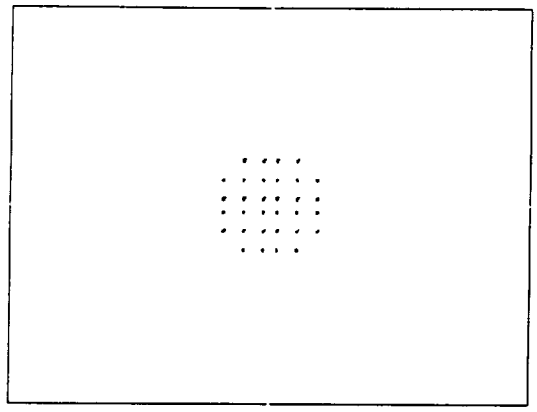
(e)



Layer 6

$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

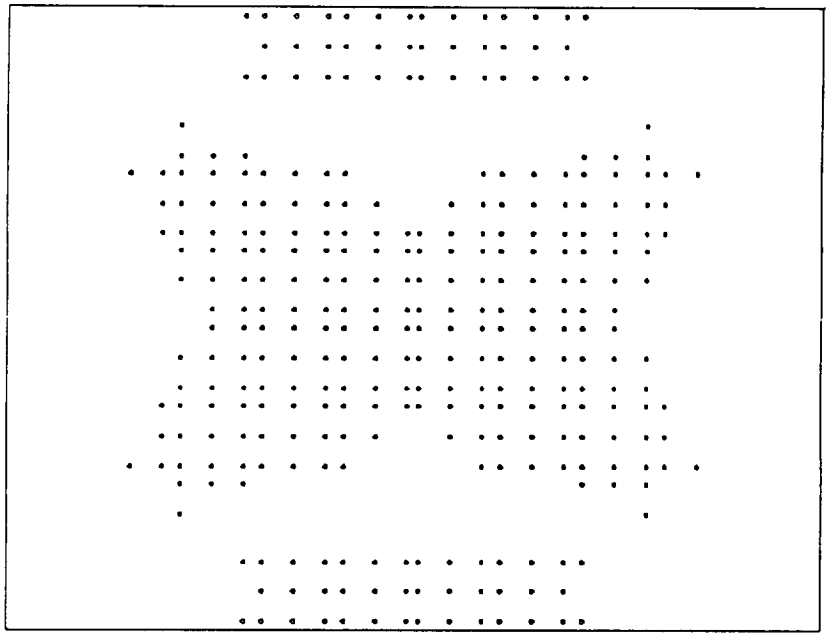
(f)



Layer 7

$[0_4/45_4/-45_4/90_4/-45_4/45_4/0_4]$

(g)



(h)

Top View of the Predicted Damage with Model using Material Degradation Method.

Top View of Damage in (a) Layer 1 (b) Layer 2 (c) Layer 3 (d) Layer 4 (e) Layer 5 (f) Layer 6 (g) Layer 7 (h) All Layers.

4.5 Observations

From the damage data obtained, and the sketches representing damage, the following observations were made:

i) The severity of damage near vicinity of the impact is maximum. Almost all of the plies were damaged in this zone.

ii) The damage seem to spread in the direction of fiber orientation, especially at the ply interfaces where the ply orientation changes.

iii) Damage is most severe in bottom layers probably due to the higher bending stresses induced. It was also observed that the first point of damage is near the vicinity of the impact and in 0° and 45° layers at the bottom interface, i.e., between layer 1 and layer 2.

iv) The damage sketches indicate that there is no evidence of damage in the 7th layer of the no-degradation model, while the 7th layer of the degradation model exhibits the damage. The reason is that the material degradation model degrades the material when the damage is predicted thus softening the material. Due to the reduced stiffness in the damage layers, the unbalanced stresses are transferred to the 7th layer causing damage in this layer, as well.

v) The damage predicted was more pervasive in the no material degradation model because no energy was dissipated during impact. In the material degradation model, material degradation resulted in energy dissipation, and caused the total system energy to be lowered.

CHAPTER 5

Conclusions

5.1 Conclusions

An analytical model was developed to predict low-velocity impact damage in graphite/epoxy composite laminates. User friendly software package was developed based on the object-oriented implementation framework called Extensible Implementation Framework for Finite Elements (EIFFE).

Four case studies were analyzed first two case studies, i.e., simply supported steel beam and isotropic and transversely isotropic plate structures, validated the model. The third case study verified the inelastic response of the model. Damage prediction in graphite/epoxy composite laminates was studied in the fourth case study, which consists of two damage models. The first model did not include material degradation, while the second model included the material degradation. Overall, the predictions obtained from the model were comparable to results obtained by Choi and Chang (1992). Based on the results obtained the following remarks can be made:

i) Damage is severe in the vicinity of impact.

ii) Damage is more prevalent in plies at the interface where the fiber orientation changes.

iii) In-ply damage grows along the fiber orientation. From this, it can be construed that damage primarily occurs in the matrix and seem to grow in the fiber direction, causing delamination.

iv) The damage size predicted by the full material degradation model is less than the damage size predicted by the model with no material degradation.

5.2 Recommendations

The model could be further improved to increase accuracy. Following are some of recommendations for improving the model:

i) Incorporate Hertz contact model to more accurately model the impact forces.

ii) Perform parametric study to determine the material degradation fraction and compare to experimental results.

iii) Explore alternative material degradation methods; one such method is reducing material stiffness within the damaged elements during the analysis.

REFEERENCES

Bahte, K. J., & Wilson, E. D. (1976). Numerical Methods in Finite Element Analysis. New Jersey: Prentice-Hall, Inc.

Belytschko, T., & Huges, J. R. (1983). Computational Methods for Transient Analysis. North-Holland: Elsevier Science Publishers B.V.

Bogdanovich, A. E., & Larve, E. V. (1992). Numerical Analyisis of Impact Deformation and Failure in Composite Plates. J. Composite Materials, 22, 520-545.

Booch, G. (1991). Object Oriented Design With Alications. RedWood, CA: The Benjamin/Cummings Publishing Company Inc.

Cantwell, W. J., & Morton, J. (1990). An Assessment of the Residual Strength of an Impact-Damaged Carbon Fiber Reinforced Epoxy. Composite Structures, 14, 303-317.

Caprino, G. (1984). Residual Strength Prediction of Impacted CFRP Laminates. J. Composite Materials, 18, 508-518.

Chaturvedi, S. K., & Sierakowski, R. L. (1985). Effect of Impactor Size on Impact Damage-Growth and Residual Properties in an SMC-R50 Composite. J. Composite Materials, 19, 100-113.

Choi, H., Wu, H. T., & Chang, F. (1991). A New Aroach toward Understanding Damage Mechanisms and Mechanics of Laminated Composites Due to Low-Velocity Impact Part I & Part II. J. Composite Materials, 25, 993-1038.

Choi, H. & Chang, F. (1992). A Model for Predicting Damage in Graphite/Epoxy Laminated Composites Resulting from Low-Velocity Point Impact. J. Composite Materials, 26, 2134-2169.

Choi, H., Wang, H., & Chang, F. (1992). Effect of Configuration and Impactor's Mass on Initial Impact Damage of Graphite/Epoxy Composite Plates Due to Line-Loading Impact. J. Composite Materials, 26, 804-827

Christoforou, A. P., Swanson, S. R., & Beckwith, S. W. (1989). Lateral Impact of Composite Cylinders. Materials: Fatigue and Fracture, Second Volume, ASTM STP 1012, Paul A. L. (Ed.). American Society for Testing and Materials, 372-386.

Crook, A. W. (1952). A Study of Some Impacts Between Metal Bodies by a Piezoelectric Method. Proceedings of the Royal Society of Mechanical Engineers, Washington, D.C., 99-107.

Daniel, I. M., & Wooh, S. C. (1990). Deformation and Damage of Composite Laminates Under Impact Loading. Impact Response and Elastodynamics of Composites, Dallas: Proceedings of Winter Annual Meeting of American Society of Mechanical Engineers, 11-26.

Dost, E. F., Ilcewicz, L. B., Avery, W. B., & Coxon, B. R. (1991). Effects of Stacking Sequence on Impact Damage Resistance and Residual Strength for Quasi-Isotropic Laminates. Fatigue and Fracture, Third Volume, ASTM STP 1110, O'Brien, T. K. (Ed.). American Society for Testing and Materials, Philadelphia, 476-500

Ekcel, B. (1993). C++ Inside & Out. Osborne McGraw-Hill, NewYork

Goldsmith, W. (1960). Impact: the Theory and Physical Behavior of Colliding Solids. London: Edward Arnold.

Gu, Z. L., & Sun, C. T. (1983). Characterization of Impact Damage in an SMRC-R50 Composite. Advances in Aerospace Structures, Materials and Dynamics, AD-06, Ed by Yuceoglu & Sierakowski, Washington, D.C.: The American society of Mechanical Engineers.

Gu, Z. L., & Sun, C. T. (1987). Prediction of Impact Damage in an SMC Composites. Composite Structures, 7, 179-190.

Gweon, S. Y., & Bascom, W. D. (1992). Damage in Carbon Fiber Composites Due to Repetitive Low-Velocity Impact Loads. J. Materials Science, 27, 2035-2047.

Husman, G. E., Whitney, J. M., & Halpin, J. C. (1975). Residual Strength Characterization of Laminated Composites Subjected to Impact Loading. Foreign Object Impact Damage to Composites, ASTM STP 568, American Society for Testing and Materials, 92-113.

Jih, C. J., & Sun, C. T. (1993). Prediction of Delamination in Composite Laminates Subjected to Low-Velocity Impact. J. Composite Materials, 27, 684-701.

Ketan, R. P., & Chang, D. C. (1983). Surface Damage of Sheet Molding Compound Panels Subject to a Point Impact Loading. J. Composite Materials, 17, 182-194.

Kook, J. S., Suzuki, M., Adachi, T., Ujihashi, S., & Matsumoto, H. (1992). Load and Strain Histories for CFRP Laminates under Low-Velocity Impact. JSME Series I, 35, 159-164.

Kubo, J. T., & Nelson, R. B. (1975). Analysis of Impact Stresses in Composite Plates. Foreign Object Impact Damage to Composites, ASTM STP 568, American Society for Testing and Materials, 228-244.

Kwon, Y. S., & Sankar, B. V. (1993). Indentation-Flexure and Low-Velocity Impact Damage in Graphite Epoxy Laminates. Journal of Composites Tecnology & Research, JCTRER, 15, 101-111.

Lin, H. J., & Lee, Y. J. (1989). On the Inelastic Impact of Composite Laminated Plate and Shell Structures. Composite Structures 14, 89-111.

Lin, H. J., & Lee, Y. J. (1990). Impact Induced Fracture in Laminated Plates and Shells. J. of Composite Materials, 24, 1179-1199.

Lloyd, B. A., & Knight, G. K. (1986). Impact Damage Sensitivity of Filament-Wound Composites Pressure Vessels. Presented at the JANNAF Meeting, New Orleans, LA.

Madan, R. C. (1991). Influence of Low-Velocity Impact On Composite Structures. Composite and Material Testing, ASTM STP 1110, O'Brien, T. K. (Ed.). Philadelphia: American Society for Testing and Materials, 457-475.

Malvern, L. E., Sun, C. T., & Liu, D. (1989). Delamination Damage in Central Impacts at Superforation Speeds on Laminated Kevlar/Epoxy Plates. Composite Materials : Fatigue and Fracture, Second Volume, ASTM STP 1012, Paul A. Lagace, (Ed.). Philadelphia: American Society for Testing and Materials, 387-405.

Moon, F. C. (1972). Wave Surface Due to Impact on Anisotropic Composite Plates. J. Composite Materials, 7, 62-79.

Moon, F. C. (1973). One-Dimensional Transient Waves in Anisotropic Plates. J. Alied Mechanics, TRANS of ASME, 95, Series E. 485-490.

Mortimer, R. W., Chou, P. C. & Carleone, J. (1975). Behavior of Laminated Composite Plates Subjected to Impact. Foreign Object Impact Damage to Composites, ASTM STP 568, American Society for Testing and Materials, 173-182.

Nguyen, D. V. (1993). Extensible Implementation Framework for Finite Elements. Software Developed at Lamar University, Texas.

Oplinger, D. W., & Slepetz, J. M. (1975). Impact Damage Tolerance of Graphite/Epoxy Sandwich Panels. Foreign Object Impact Damage to Composites, ASTM STP 568, American Society for Testing and Materials, 30-48.

Poe, C. C. (1991). Relevance of Impactor Shape to Nonvisible Damage and Residual Tensile Strength of a Thick Graphite/Epoxy Laminate. Composite Materials : Fatigue and Fracture, ASTM STP 1110., Philadelphia: American Society for Testing and Materials, 501-527.

Poe, C. C. Jr., & Illg, W. (1986). Tension Strength of a Thick Graphite/Epoxy Laminate After Impact By a 1-2 in Radius Impactor. NASA-TM 87771.

Poe, C. C. Jr., & Garber, D. P. (1987). Strength of Thick Graphite/Epoxy Rocket Motor Case After Impact By a Blunt Object. NASA-TM 89099.

Preston, Jr., & Cook, T. S. (1975). Impact Response of Graphite-Epoxy Flat Laminates Using Projectiles That Simulate Aircraft Engine Encounters. Foreign Object Impact Damage to Composites, ASTM STP 568, American Society for Testing and Materials, 49-71.

Reddy, J. N. (1984).. An Introduction to the Finite Element Method. New York: McGraw-Hill International Editions.

Reddy, J. N. (1983). Dynamic (Transient) Analysis of Layered Anisotropic Composite Material Plates. International Journal for Numerical Methods in Engineering, 19, 237-255.

Sjöblom, P. O., Hartness, J.T., & Cordell, T. M. (1988). On Low-Velocity Impact Testing of Composite Materials. J. Composite Materials, 22, 30-52.

Shivakumar, K. N., Elber, W., & Iiig, W. (1985). Prediction of Impact Force and Duration Due to Low-Velocity Impact on Circular Composite Laminates. ASME J. Alied Mechanics, 52, 674-680.

Strait, L. H., Karasek, M. L., & Amateau, M. F. (1992). Effect of Stacking Sequence on Impact Resistance of Carbon Fiber Reinforced Thermoplastic Toughened Epoxy Laminates. J. Composite Materials, 26, 1725-1740.

Stroustrup, B. (1993). The C++ Programming Language. Addison-Weseley Publishing Company, NY.

Sun, C. T., & Chattopadhyay, S. (1985). Dynamic Response of Anisotropic Laminated Plates Under Initial Stress to Impact of a Mass. ASME J. Alied Mechanics, 52, 693-698.

Sun, C. T. (1977). An Analytical Method for Evaluation of Impact Damage Energy of Laminated Composites. Composite Materials: Testing and Design (Fourth Conference), ASTM STP 617, American Society for Testing and Materials, 427-440.

Sun, C. T., & Chen, J. K. (1985). On the Impact of Initially Stressed Composite Laminates. J.Composite Materials, 19, 490-504.

Sun, C. T., & Jih, C. J. (1990). Mechanics of Delamination in Composite Laminates Subjected to Low-Velocity Impact. On Impact Response and Elastodynamics of Composites, Mal, A.K & Rajapakse, Y.D.S, eds ASME ADM, 116, 1-10.

Tan T. H., & Sun C. T. (1985). Use of Statical Indentation Laws In The Impact Analysis of Laminated Composite Plates. J. Alied Mechanics, 52, 6-12.

Tian, Z., & Swanson, S. R. (1992). Residual Strength Prediction on a Ply-by-Ply Basis for Laminates Containig Impact Damage. J. Composite Materials, 26, 1193-1206.

Tsai, W. S. (1971). A General Theory of Strength for Anisotopic Materials. J. Composite Materials, 5, 58-80.

Willis, J. R. (1966). Hertzian Contact of Anisotropic Bodies. Journal of Mechanics and Physics of Solids, 14, 163-176.

Wu, H. T. (1986). Impact Damage of Composites. Phd Dissertation, Department of Aeronautics and Astronautics, Stanford University.

Wu, H. T., & Springer, G. S. (1986). Impact Damage of Composites. Proceedings of The American Society For Composites, First Technical Conference, Technomic Publishing Co, Inc., Dayton, Ohio.

Wu, H. T., & Springer, G. S. (1988). Impact Induced Stresses, Strains and Delaminations in Composite Plates. J. Composite Materials, 22, 533-559.

Wu, H. T., & Springer, G. S. (1988). Measurement of Matrix Cracking and Delamination Caused by Impact on Composite Plates. J. Composite Materials, 22, 518-532.

Wu, H. T., & Chang, F. K. (1989). Transient Dynamic Analysis of Laminated Composite Plates Subjected to Transverse Impact. Computers and Structures, 3, 453-466.

Yang, T. Y., & Sun, C. T. (1973). Finite Element for the Vibration of Framed Shear Walls. J. Sound and Vibrations, 27, 297-311.

Yang, S. H., & Sun, C. T. (1982). Indentation Law for Composite Laminates. Composite Materials: Testing and Design, ASTM STP 787, American Society for Testing and Materials, 425-429.

Yener, M., & Wolcott, E. (1989). Damage Assessment Analysis of Composite Pressure Vessels Subjected to Random Impact Loading. Journal of Pressure Vessel Technology, Transactions of ASME, 111, 124-129.

Zienkiewicz, O. C. (1977). The Finite Element Method. Edn 3rd, NewYork: McGraw-Hill.

Appendix

Program Listing

```
//=========================================================================
//    Module:           COMPOSITE.H
//    Description:      Interface file for Composite Element classes
//=========================================================================

#ifndef __COMPOSITE_H
#define __COMPOSITE_H

#include <afx.h>
#include <feobj.h>
#include <structrl.h>
#include <material.h>
#include <fstream.h>

_CLASSDEF (RKC_3DCompositeTemplate)
_CLASSDEF (RKC_CompositeElement)
_CLASSDEF (RKC_3DTemplate)


class _CLASSTYPE RKC_3DTemplate:public VN_3DTemplate
{
      public:
         RKC_3DTemplate (const CString & Name, RVN_Material Material,
            PVN_FunctionArray pSFArray=NULL,
         PVN_Integrator pKIntegrator=NULL, PVN_Integrator
            pInitialFIntegrator=NULL,
         PVN_Integrator pBodyFIntegrator=NULL, PVN_Integrator
            pMIntegrator=NULL);
         virtual dataType ComputeStrainEnergy(RVN_StructuralElement
            rElement, RCVN_Vector rElementDisplacements);
         PVN_Vector GetCoordinatesAt(RVN_StructuralElement rElement,
            RCVN_DataArray X);

      protected:
         RKC_3DTemplate() {};
         PVN_Vector GetCoordinatesAt(RCVN_DataArray X);
};


enum degradationType{COMPLETE,PARTIAL,NOCOMPLETE};

class _CLASSTYPE RKC_3DCompositeTemplate : public RKC_3DTemplate
{
DECLARE_SERIAL (RKC_3DCompositeTemplate)
      public:
         RKC_3DCompositeTemplate (const CString & Name, RVN_Material
            Material,PVN_Integrator pLKIntegrator,PVN_FunctionArray
            pSFArray=NULL,
```

```
            PVN_Integrator pKIntegrator=NULL, PVN_Integrator
               pInitialFIntegrator=NULL,
            PVN_Integrator pBodyFIntegrator=NULL, PVN_Integrator
               pMIntegrator=NULL);
            ~RKC_3DCompositeTemplate();
            RVN_SymMatrix GetE (dataType z);
            RVN_SymMatrix GetE();
            indexType GetLayerNumAt (dataType z)const;
            indexType GetLayerNumAt(indexType PointNo)const;
            virtual indexType NumLayerIntegPts()const;
            virtual PVN_SymMatrix ComputeStiffnessMatrix ();
            PVN_SymMatrix ComputeStiffnessMatrixAtPoint(indexType PointNo);
            virtual PVN_Vector ComputeAverageInitialForceVector
               (RCVN_Vector ElementDisplacments);
            virtual PVN_Vector ComputeAverageStressesAtPoint(RCVN_Vector
               ElementDisplacments,indexType LocalPointNo,indexType
               FrmLayerNo, indexType ToLayerNo);
            virtual PVN_Vector ComputeInitialForceVector (RCVN_Vector
               ElementDisplacements);
            PVN_Vector ComputeStressesAt (RCVN_Vector ElementDisplacements,
               RCVN_DataArray X);
            void GetIntrinsicCoordinatesAt(RVN_DataArray X, indexType
               PointNo) const;
            BOOL UpdateElementCondition(RCVN_Vector
               ElementDisplacments,degradationType eDegrade);
            void Serialize(CArchive &ar);

         protected:
            dataType prevOrientation, m_dStrainEnergy;
            PVN_SymMatrix m_pEp;
            PVN_NdxArray m_pMarksArray;
            PVN_Matrix m_pGaussPtsWtsMatrix;
            PVN_CubeIntegrator m_pLKIntegrator;
            void CreateGaussPtsWtsMatrix();
            void CreateMarksArray();
            virtual void ComputeE();

            RKC_3DCompositeTemplate();
            RKC_3DCompositeTemplate(RCRKC_3DCompositeTemplate );
            RCRKC_3DCompositeTemplate operator=(RCRKC_3DCompositeTemplate
               );
};


class _CLASSTYPE RKC_CompositeElement : public VN_StructuralElement
{
DECLARE_SERIAL (RKC_CompositeElement)
      public:
         RKC_CompositeElement (RRKC_3DCompositeTemplate
            Template,PVN_NodeArray pNodeArray,
         RVN_DataArray rThicknessArray, RVN_DataArray
            rOrientationArray);
         ~RKC_CompositeElement();
```

```
        indexType NumLayers() const;
        dataType Thickness()const;
        dataType LayerThicknessAt (indexType LayerNo) const;
        dataType LayerOrientationAt (indexType LayerNo) const;
        dataType LayerLowerCoordinateAt (indexType LayerNo)const;
        dataType LayerUpperCoordinateAt (indexType LayerNo)const;
        dataType LayerThicknessRatioAt (indexType LayerNo) const;
        dataType LayerMidZCoordinateAt (indexType LayerNo)const;
        dataType LayerMidEtaCoordinateAt (indexType LayerNo) const;
        indexType GetLayerNumAt (dataType z)const;
        dataType GetIntrinsicCoordinate (dataType z)const;
        dataType GetExtrinsicCoordinate (dataType Eta)const;
        dataType TransformCoordinate (dataType CoordX, dataType
           LowerLimitX, dataType UpperLimitX, dataType LowerLimitx,
           dataType UpperLimitx) const;
        PVN_SymMatrix ComputeStiffnessMatrixAtPoint(indexType PointNo)
           const;
        void ComputeAverageInitialForceVector(RCVN_Vector
           ElementVector);
        RVN_Vector GetAverageInitialForceVector() const;
        virtual void ComputeInitialForceVector(RCVN_Vector
           ElementVector);
        void RemoveAverageInitialForceVector();
        BOOL UpdateElementCondition(RCVN_Vector ElementDisplacements,
           degradationType eDegrade = NOCOMPLETE);
        void UpdateStiffnessMatrix(dataType DegradationFactor = 1.0);
        BOOL GaussPtStsOK(indexType PointNo) const;
        void SwitchAllGaussPtsSts(BOOL bFlag);
        void SwitchGaussPtSts(indexType PointNo, BOOL bFlag);
        indexType GaussPtReport(indexType PointNo) const;
        void SwitchGaussPtRptSts(indexType PointNo, indexType Flag);
        void SwitchAllGaussPtsRptSts(indexType Flag);
        RVN_SymMatrix GetE()const;
        virtual RVN_SymMatrix GetE (dataType z)const;
        void PrintElementCondition(ofstream &os, indexType
           ElementNo,indexType iter);
        void Serialize(CArchive &ar);
    protected :
        indexType m_uNumLayers,m_iKIntegPts,m_iLKIntegPts;
        dataType m_dThickness,m_dLowerLimit,m_dUpperLimit;
        PVN_DataArray m_pThickArray, m_pOrientationArray;
        PVN_NdxArray m_pbGaussPtsPrntStsArray;
        BOOL *m_pbGaussPtsStsArray;
        dataType ComputeThickness();
        void CreateGaussPtsStsArray();

        RKC_CompositeElement();
        RKC_CompositeElement (RCRKC_CompositeElement );
        RCRKC_CompositeElement operator=(RCRKC_CompositeElement );
};
#include "composit.inl"
#endif
```

```
//=========================================================================
//    Module:                  INITIALC.H
//    Description:       Interface file for Intial
//                       Boundary Condition class
//=========================================================================


#ifndef _INITIALC__H
#define _INITIALC__H
#include <afx.h>
#include <datadefs.h>
#include <matrix.h>
#include <feobj.h>

_CLASSDEF (RKC_InitialC)

class _CLASSTYPE RKC_InitialC : public CObject
{
      public:
         RKC_InitialC(RVN_Node rNode, dataType Value, UINT dir);
         RVN_Node Node() const;
         dataType Value() const;
         UINT Direction() const;
         PRKC_InitialC CopyTo (RVN_Node newNode);
         void Apply (RVN_Vector F) const;
         virtual void printOn (ostream & os) const;
         protected:
         RKC_InitialC();
         PVN_Node m_pNode;
         dataType m_dValue;
         UINT m_iDirection;
      private:
         RKC_InitialC (RCRKC_InitialC);
         RCRKC_InitialC operator = (RCRKC_InitialC);
};
#include "InitialC.inl"
#endif
```

```
//=========================================================================
//    Module:           TRNSCOLLS.H
//    Description:      Interface file for Transient
//                      Boundary Condition Collection classes
//=========================================================================

#ifndef __TRNSCOLL_H
#define __TRNSCOLL_H
#include <fecolls.h>
#include "initialc.h"

_CLASSDEF(RKC_InitialCArray)
_CLASSDEF (RKC_InitialCList)

class _CLASSTYPE RKC_InitialCArray : public VN_ObjectArray
{
      public:
         RKC_InitialCArray (BOOL bOwnElements = FALSE) : VN_ObjectArray
            (bOwnElements) {};
         RRKC_InitialC IC(int i) { return (RRKC_InitialC) *ElementAt(i);
            }
         int AddIC (PRKC_InitialC pIC) { return Add ((CObject *) pIC); }
         void SetICAt (int i, PRKC_InitialC pIC) { SetAt (i, (CObject *)
            pIC); }
         void InsertICAt (int i, PRKC_InitialC pIC, int nCount = 1)
         { InsertAt (i, (CObject *) pIC, nCount); }
         void SetICAtGrow (int i, PRKC_InitialC pIC) { SetAtGrow (i,
            (CObject *) pIC); }
};

class _CLASSTYPE RKC_InitialCList : public VN_ObjectList
{
      public:
         RKC_InitialCList (BOOL bOwnElements = TRUE) : VN_ObjectList
            (bOwnElements) {};
         POSITION AddHeadIC (PRKC_InitialC pIC) { return AddHead
            ((CObject *) pIC); }
         POSITION AddTailIC (PRKC_InitialC pIC) { return AddTail
            ((CObject *) pIC); }
         POSITION InsertICAfter (POSITION pos, PRKC_InitialC pIC) {
            return InsertAfter (pos, (CObject *) pIC); }
         POSITION InsertICBefore (POSITION pos, PRKC_InitialC pIC) {
            return InsertBefore (pos, (CObject *) pIC); }
         void SetICAt (POSITION pos, PRKC_InitialC pIC) { SetAt (pos,
            (CObject *) pIC); }
         PRKC_InitialC RemoveHeadIC() { return (PRKC_InitialC)
            RemoveHead(); }
         PRKC_InitialC RemoveTailIC() { return (PRKC_InitialC)
            RemoveTail(); }
         RRKC_InitialC IC (POSITION pos) { return (RRKC_InitialC)
            *GetAt(pos); }
         RRKC_InitialC HeadIC() { return (RRKC_InitialC) *GetHead(); }
         RRKC_InitialC TailIC() { return (RRKC_InitialC) *GetTail(); }
```

```
        RRKC_InitialC NextIC (POSITION & pos) { return (RRKC_InitialC)
           *GetNext(pos); }
        RRKC_InitialC PrevIC (POSITION & pos) { return (RRKC_InitialC)
           *GetPrev(pos); }
};
#endif


//=========================================================================
//    Module:           TRANSSOL.H
//    Description:      Interface file for Transient Model class
//=========================================================================

#ifndef __TRANSSOL_H
#define __TRANSSOL_H

#include <afx.h>
#include <feobj.h>
#include <structrl.h>
#include <material.h>
#include"trnscoll.h"
#include "composit.h"
#include <fstream.h>

_CLASSDEF (RKC_TransientModel)

enum approachType{DIRECTVECTOR,DEFAULT};
enum forceType{AVERAGEFORCES,DEFAULTFORCES};

class _CLASSTYPE RKC_TransientModel:public VN_StructuralModel
{
      public:
        RKC_TransientModel(const CString & Name=" ");
        ~RKC_TransientModel();
        virtual void Flush();
        void AddInitialC(PRKC_InitialC pIC);
        int NumInitialCs()const;
        void ComputeAverageElementInitialForces();
        virtual void InitiateImpulse(dataType dMass,approachType
            eAppType = DEFAULT,forceType eForceType =
            DEFAULTFORCES,dataType Alpha =0, dataType Beta = 0);
        virtual void Solve(dataType Incr);
        virtual void ComputeNodalAccelarations(RVN_Vector
            rAccelarations, RCVN_Vector InvM,RCVN_Vector F, RCVN_Vector
            IVel);
        virtual void ComputeNodalAccelarations(RVN_Vector
            rAccelarations,RCVN_Vector InvM, RCVN_Vector F);
        virtual void UpdateNodalDisplacements(RVN_Vector
            rSolution,RCVN_Vector IVel, RCVN_Vector Acc, dataType
            DeltaTm);
        virtual void UpdateNodalVelocities(RVN_Vector rVelocites,
            RCVN_Vector rAcc, dataType DeltaTm);
```

```
        dataType ComputeCriticalTime(dataType TOL = 1e-7, indexType
            Iter = 2000);
        dataType ComputeSystemKineticEnergy();
        dataType ComputeSystemStrainEnergy();
        RVN_Vector NodalVelocities()const;
        RVN_Vector NodalForces()const;
        RVN_Vector NodalAccelarations()const;
        virtual void UpdateModel();
        BOOL UpdateElementsCondition(degradationType eDegrade =
            NOCOMPLETE,dataType DegradationFactor=1);
        void SerializeVectors(CArchive & ar);

    protected:
        RKC_InitialCList m_InitialCList;
        PVN_Vector m_pIVel,m_pAcc, m_pInvMass,
            m_pGlobalInitialForceVector,m_pGlobalKXVector,
            m_pGlobalCXVector, m_pGlobalForceVector, m_pMassVector;
        approachType m_eAppType;
        forceType m_eForceType;
        dataType m_dAlpha, m_dBeta;
        ofstream os;
        BOOL m_bCrTmComputed;
        void ApplyEssentialBCs (RVN_Vector x, applyType Flag);
        void AssembleMassVector();
        void ApplyInitialConditions(RVN_Vector Value,dataType Mass);
        void ApplyNaturalBCs(RVN_Vector Value);
        void AssembleElementInitialForces();
        void ComputeAssembleElementKXVectors(RCVN_Vector
            ElementDisplacments);
        void ComputeAssembleElementCXVectors(RCVN_Vector
            ElementVelocityVector);
        void UpdateElementInitialForces();
        void UpdateElementStiffnesses(dataType DegradationFactor=1);
        void UpdateForceVector();
        void InvertMassVector();
        void CreateVectors();
        void RemoveVectors();
        void RemoveElementStiffnesses();
        private:
        RKC_TransientModel(RCRKC_TransientModel);
        RCRKC_TransientModel operator=(RCRKC_TransientModel);
};

#include "transsol.inl"
#endif
```

```
//=========================================================================
//    Module:           MODEL.H
//    Description:      Interface file for Conversion Module From
//                      Nastran Data File to EIFFE Objects
//=========================================================================
#ifndef _MODEL_H
#define _MODEL_H

#include <afx.h>
#include <afxwin.h>
#include <ostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iomanip.h>
#include <feobj.h>
#include <structrl.h>
#include "nasdef.h"
#include "composit.h"
#include "transsol.h"
#include "plancomp.h"


_CLASSDEF (RKC_Model)

class _CLASSTYPE RKC_Model
{
      public:
         RKC_Model(modelType eModelType, solutionType eSolType, CString
            ProblemHeader);
         ~RKC_Model();
         PVN_ElementTemplate CreateElementTemplate(modelType eModelType,
            elementType eElementType, materialType eMat );
         PVN_StructuralTemplate
            CreateStructuralElementTemplate(elementType eElementType,
            materialType eMat);
         PVN_Element CreateElement(RVN_ElementTemplate pElementTemplate,
            PVN_NodeArray pConNodeArray, materialType eMat);
         PVN_StructuralElement
            CreateStructuralElement(RVN_StructuralTemplate
            pElementTemplate, PVN_NodeArray pConNodeArray, materialType
            eMat);
         void SolveStaticModel();
         void SolveTransientModel();
         void ReadinNastranModel(ifstream _FAR & os, CString &
            MatFilename);
         void PrintNodal(RVN_Vector rVector, CString Item,ostream _FAR
            &os, indexType Case = 1,printFormatType
            eFormat=SPECIAL)const;
         void PrintStresses(ostream _FAR &os, indexType Case= 1,
            printFormatType eFormatType = SPECIAL) const;
         RVN_Vector Solution();
```

```
    void GetViewClass(CView *pView);
    void ComputeCriticalTimeStep();
    dataType ComputeElementStrainEnergy(indexType ElementNo);

protected:
    CString m_sName;
    indexType m_uNumIterations;
    UINT m_iDOFNum;
    PVN_Vector m_pSolutionVector,m_pVelVector, m_pAccVector,
       m_pForVector;
    CView *m_pView;
    PVN_FunctionArray m_pFunctionArray;
    PVN_Integrator m_pKIntegrator, m_pInitialFIntegrator,
       m_pBodyFIntegrator,m_pMassIntegrator,m_pLKIntegrator;
    PVN_StructuralModel m_pFEModel;
    PVN_ElementTemplate m_pElementTemplate;
    PVN_Material m_pMaterial;
    indexType m_iPrevElement;
    indexType m_iElementID,m_iNodeNo,m_iStressElementID,m_iLayerNo;
    dataType m_dDeltaTm,
       m_dCrTm,m_dStrainEnergy,m_dKineticEnergy,m_dModelTime;
    modelType m_eModelType;
    solutionType m_eSolutionType;
    elementType m_eElementType;
    materialType m_eMaterialType;
    PVN_NodeArray m_pNodeArray;
    PVN_ElementArray m_pElementArray;
    PVN_ElementTemplateArray m_pElementTemplateArray;
    PVN_ObjectArray
       m_pNdxContainer,m_pOrDataContainer,m_pThDataContainer;
    PVN_NdxArray m_pStressElementArray;
    ofstream
       osdisp,osvel,osforce,osacc,osini,osdam,oscomb,osstress;
    void FirstScan(ifstream _FAR & is);
    virtual void ReadGrid(ifstream _FAR & is);
    virtual void ReadElement(ifstream _FAR & is, CString &
       MatFilename);
    virtual void SetSizeofArrays();
    virtual void ReadBoundaryConditions(ifstream _FAR
       &is,boundaryType eBoundary);
    virtual void ReadElementProperties(const CString & sMaterial,
       CString & MatFilename);
    virtual PVN_Element CreateShellElement(RVN_UINTArray iConnData,
       materialType eMat);
    virtual PVN_Element Create8NodeBrickElement(RVN_UINTArray
       iConnData,materialType eMat);
    virtual PVN_Integrator CreateKIntegrator(elementType eElement);
    PVN_Integrator CreateInitialBodyFIntegrator(elementType
       eElement);
    PVN_Integrator CreateBodyFIntegrator(elementType eElement);
    PVN_Integrator CreateMassIntegrator(elementType eElement);
    PVN_Integrator  CreateLayerKIntegrator(elementType eElement);
```

```
        PVN_FunctionArray CreateShapeFunctionArray(elementType
           eElement);
        void CreateIntegrators(elementType eElement);
        virtual PVN_3DTemplate Create3DTemplate(materialType eMat);
        virtual PVN_PlaneStressTemplate CreatePlaneStressTemplate
           (materialType eMat);
        const char* ProcessString(CString _FAR &var);
        virtual void ReadLabel(istream _FAR & is, const char* Label);
        void CreateOutputFiles();
        void CloseOutputFiles();
        void PrintAllVectors(indexType EveryIter=1);
        void Print(indexType NodeNo, RVN_Vector Displacements,
           RVN_Vector Velocities, RVN_Vector Accelarations, RVN_Vector
           Forces,ofstream _FAR & os, indexType Case )const;
        void PrintStressesAt(indexType ElementNo, RCVN_Vector X,
           RCVN_Vector rStressVector,RCVN_Vector rStrainVector,ofstream
           _FAR & os, indexType Case)const;
        void PrintElementsStatus(ofstream &os, indexType iter);
        PVN_Vector ComputeElementStressesIn(RVN_StructuralElement
           rElement, RVN_DataArray X,indexType LayerNo=0);
};
#endif
```